Juhani Lavonen

Developing a Web Application for Indoor Map Data Management

Metropolia University of Applied Sciences

Bachelor of Engineering

Degree Programme

Thesis

7 May 2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Juhani Lavonen Web-sovellus sisätilakarttojen hallintaan 53 sivua + 14 liitettä 7.5.2018
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	tieto- ja viestintätekniikka
Ammatillinen pääaine	Mobile Solutions
Ohjaajat	teknologiajohtaja Mikko Virkkilä lehtori Olli Alm

Insinöörityön tavoitteena oli luoda tehokas ja käyttäjäystävällinen sovellus sisätilakarttadatan hallintaan. Työ tehtiin yritykselle, joka on erikoistunut sisätilapaikannukseen ja karttoihin. Insinöörityössä lähestyttiin sisätilakarttojen hallintaa käyttäjäkokemuksen ja geospatiaalisen datan hallinnan näkökulmista.

Työssä tehtiin kaksi kattavaa tutkimusta yrityksen työntekijöiden keskuudessa. Ensimmäinen tutkimus suoritettiin työn alkuvaiheessa, ja siinä kerättiin käyttäjävaatimuksia. Sovelluksen kehitystyö alkoi tutkimustulosten analysoinnin jälkeen. Toinen tutkimus oli käyttäjätestaus, joka tehtiin sovelluksen sisäisen julkaisun aikana.

Saatujen tulosten ja käyttäjätestauksen palautteiden perusteella todettiin, että insinöörityössä tehty sovellus täytti sille asetetut vaatimukset. Työntekijät kokivat karttojen ylläpito- ja hallintaprosessin tehokkaaksi ja käyttäjäystävälliseksi.

Avainsanat	sisätilakartat, web-sovellus, GeoJSON, Mapbox, web-kehitys, käyttäjäkokemus

Author	Juhani Lavonen
Title	Developing a Web Application for Indoor Map Data Management
Number of Pages	53 pages + 14 appendices
Date	7 May 2018
Degree	Bachelor of Engineering
Degree Programme	ICT
Professional Major	Mobile Solutions
Instructors	Mikko Virkkilä, CTO Olli Alm, Senior Lecturer

The goal of this thesis was to create an effective and user-friendly application for managing indoor map data. The work was done for a company called Steerpath. This thesis approaches the indoor map data management from two perspectives: from the point of view of user experience and how to manage geospatial data in the most efficient way.

During this study two comprehensive surveys were conducted among the employees of Steerpath. The first survey was conducted at the beginning of the project and it was about gathering user requirements for the product. After analysing the results of the survey, the development of the product began. The second survey was a usability test which was held at the time of internal release of the developed application.

Based on the results and feedback from the usability test, the application was proven to meet the requirements that were set for it. The employees felt that the application makes the map maintenance and management process more user friendly and efficient.

Keywords	indoor map, web application, geospatial data, GeoJSON, Mapbox, web development, editor, user experience, usability

Metropolia

Contents

List of Abbreviations

API	Application Programming Interface. A collection of protocols which enable communication between different software.
CAD	Computer-aided design. A system used to help with creation of designs. These designs can be blueprints of buildings or parts of machine.
DOM	Document Object Model. A way to describe a standard way for communicating with document on the web.
GIS	Geographic Information System. A technology which enables storing and managing geographical data.
IPS	Indoor Positioning System. A system which enables locating and wayfinding with high accuracy indoors.
POI	Point of Interest. A map marking which represents either an abstract or physical point of interest in the map such as restaurant or attraction.
REST	Representational State Transfer. A model of a web service which provides an ability to exchange and use information between computer systems on the internet.

1 Introduction

As our world becomes more automated and the demand for time-efficiency increases, the need for indoor location services has risen during the past few decades. When visiting a large and complex indoor environment, people need to know where they are in a room, where the room is on a floor and which floor they are on within the building. (Goswami, 2013.) Indoor location and wayfinding services typically require an indoor map in order to show the user's location and surroundings. Indoor maps represent a detailed cartography of the building containing key elements such as walls, floor, rooms, and other Points of Interest (POI). This thesis focuses on indoor maps and how they can be managed. The thesis was done for a Finnish company called Steerpath, which specializes in delivering both indoor location and indoor map services. The goal of this project was to enhance the company's indoor map maintenance and update services.

Part of Steerpath indoor map creation is extracting data from existing digitalised drawings of the maps. These drawings are usually in a Computer-Aided Design (CAD) format. Edited CADs are processed through Steerpath map pipeline and the end result is detailed indoor map presented in vector format. Should there be any changes in the building these updates need to be added to the original drawing which will then get reprocessed through the system. There was a clear need within Steerpath to improve the current system and web application was explored as one of the solutions. The main goal was to make the editing process more efficient and user friendly. In this thesis the application that was developed is referred as Indoor Map Editor.

Creating an efficient web based indoor map editor and making the user experience fluid and intuitive were two main points of the thesis. The efficiency of the editor can be measured by how well it performs on editing geospatial data. This thesis will first present the concepts of digitalised maps, Steerpath's map data models and current state of the maintenance process in chapter 2. Chapter 3 reviews the User Experience designs of existing geospatial editors. In order to fully understand the requirements for the Indoor Map Editor a comprehensive user requirement survey was conducted among the employees of Steerpath. The answers of the survey are analysed in chapter 4 and the architecture and implementations of these requirements are explained in chapter 5 and 6. To measure the success of the application, a usability test was carried out, and the results of the test are analysed in chapter 7.

2 Indoor Maps

Whereas outdoor maps are more focused on showcasing larger scale objects such as cities, road, rivers and oceans indoor maps are focused on the map data inside the buildings. Indoor map data tends to change more often, and it is more critical to businesses than outdoor map data when integrated into their systems and services. It is also more dense and detailed and is visualized in deeper zoom levels. Showing more data more densely can also make indoor maps more complex. The complexity of indoor maps can also be seen in the vertical axis: buildings can have hundreds of floors layered on top of each other whereas outdoor maps don't have as much data on the vertical axis. (Haveri et al., 2012) The aim of this thesis is to find a way to manage indoor map data in efficiently and user-friendly way.

2.1 Geographical Information System

Geographical Information Systems (GIS) is a technology which is used to store geographical data which is usually coupled with contextual attributes. Attributes can contain additional information about each of the spatial feature. For example, in addition to building's spatial data there may exist contextual attributes such as type and number of floors related to the building. (Dempsey, 2018.)

The concept of coupling spatial data with contextual attributes can also be used in indoor maps. In addition to rooms geographical data, attributes such as room's name and category can be stored in the system. Steerpath indoor map data follows the GIS practises by having contextual attributes stored in addition to geometric data. An evident use of GIS technology is using it in computer cartography. Traditionally there are two ways of representing GIS data: raster images and vectors features. (Dempsey, 2018.)

2.2 Map Presentation

Raster tiled maps can be thought as matrixes where each of the cells represent a part of the map as a digitalised image. These images can be referenced as tiles. At the highest zoom level, entire world map can be presented in one tile and as the map gets zoomed in deeper tiles will be split into four pieces. This means that raster maps require relatively

more data as each time the map gets zoomed in new set of tile images is required. (Ordnance Survey, 2018.)

The main difference between raster tiles and vector tiles is that vector tiles store the geometrical and contextual information of each map feature, while a raster tile store the visualised flattened compound of the data. The features in the vector maps can be:

- Points, which can be seen as symbols or labels indicating for example POIs or area name. Points consist of only one pair of latitude and longitude coordinates.
- Lines, which can represent roads or rivers. Lines may have multiple pairs of coordinates.
- Polygons which can be used to visualise oceans or land masses. Polygons coordinate pairs start and end at the same point. Polygons can be used to represent area associated with the POI.

As the maps gets zoomed in these vector features scale concurrently enabling high quality maps at all zoom levels (Ordnance Survey, 2018.) Vector tile features might also have contextual attributes stored in it as metadata. These attributes can be used to group and categorise the map data. Compared to raster map presentations vector maps provide more fluid, responsive and less memory consuming maps. (Ordnance Survey, 2018.)

2.3 Map Data Formats

GeoJSON is geospatial data format which is based on JavaScript Object Notation (JSON) and it can be used to define different types of JSON objects which are used to represent geographic features, their properties and spatial coordinates. Traditionally coordinates are presented as longitudes and latitudes. There are seven types of geometries that GeoJSON supports:

- **Points** are the simplest type geometry and they represent a single position.
- **LineStrings** represent a line which requires starting and end point.
- **Polygons** have the same first and end coordinate.
- **Multipoints** are used to represent a collection of single points.
- **MultiLineStrings** contain collection of single lines.
- **MultiPolygons** represent collection of polygons.

- **GeometryCollections** can contain multiple different type of geometries.

In addition to geometric data GeoJSON objects may contain contextual attributes. Objects that contain both spatial data and contextual attributes are referred as Features and collection of Features is called FeatureCollection. (MacWright, 2015.) Example of how GeoJSON Feature representing one POI can be encoded is shown in the Listing 1 below.

```
{
      "type": "Feature",
       "properties": {
            "css_class": "café_poi",
            "buildingRef": "67",
            "layerIndex": 2,
            "title": "Café1"
      },
       "geometry": {
            "type": "Point",
            "coordinates": [
                  24.825196266174316,
                  60.216375997829886
            ]
      }
}
```

Listing 1. Example of GeoJSON Feature where contextual attributes are stored in the properties object and spatial data in geometry object. The “css_class” attribute determines the type of the POI. Attributes “buildingRef” and “layerIndex” are used to describe to which building and floor the POI belongs to.

GeoJSON features are not bound to represent a physical structure which means that any geographical space can be a GeoJSON feature. The concepts of GeoJSON are based on pre-existing open GIS standards but they are modified to fit web application development. (Butler et al., 2016.)

Steerpath indoor map vector data is encoded in GeoJSON. For example, a POI representing a building wing can be thought as GeoJSON Feature with abstract geographical space. While GeoJSON format can be used with indoor maps there exists some concepts that are not natively supported by GeoJSON. For example, the concept of building’s floor and how they can be layered on top of each other is not natively supported in GeoJSON.

TopoJSON is a geospatial data format created based on GeoJSON to tackle its complexity (Macwright, 2015.). Instead of listing geometric features TopoJSON uses topology for storing data. Topology means that coordinates have indexes to which

features refer to. Topological encoding eliminates redundancy since geometries can be shared and utilised by multiple features. For instance, coordinates of polygon object can be used for both fill and outline representation. Therefore, TopoJSON is more compact compared to GeoJSON encoding (Bostock, 2016.). While TopoJSON has advantages over GeoJSON due to its compactness it is not as widely supported by existing geographical tools and libraries as GeoJSON. Therefore, it was not considered to be used in the project as it would require work to modify existing tools to support it. (Virkkilä, 2018.)

2.4 Comparing Existing Map Rendering Solutions

The application developed in this project for managing the indoor map data requires web map library in order to show the maps. In addition to showing the map data on the web the library should also provide tools for managing and editing the geometric data. There exist open source solutions for visualising map data on the web but in order to find the most suitable solution for this project the alternatives needed to be reviewed and analysed.

GoogleMaps API provides multiple map services such as geolocation, autocomplete boxes and traffic data in addition to a map presentation and tile hosting. These services are available for free when there are low number usages. While GoogleMaps API provides efficient rendering for as many as over 10 000 markers the interactions however customisations of these markers are limited. Also, GoogleMaps API does not support changing completely the visual look of their maps or the feature's that its representing (Temprano, 2016.). Google Maps doesn't support hosting the map tiles outside of their servers which can be seen as one of the disadvantages as well compared to the other map rendering solutions.

Leaflet is a tile-based open source JavaScript mapping library developed by Cloudmade. Leaflet supports both raster and vector tiles in rendering and also Vector Mark-up Language (VML) for older versions of Internet Explorer. Since Leaflet is a raster tile-based library the tiles are hosted and cached on the server and displayed on the client (Taraldsvik, 2011.). Much like some of the other compared libraries, Leaflets advantages are that it is supported across many platforms and that it's open source. One of the advantages Leaflet has is that it has extensive drawing tools, but the raster presentation

of the maps makes it unsuitable solution for the application since the company's map data is encoded in vector format.

OpenLayers is an open source map library that allows viewing dynamic maps for web use. OpenLayers supports raster and vector tiles and due it's open source nature tiles can be rendered from different sources. OpenLayers map consists of layers, view, interactions and controls. Layers can be thought as containers which show data from sources. View manages map's viewport attributes such as centre points coordinates, rotation, projection and resolution. Interactions can be used to alter the map content and they vary from simple dragging and panning interactions to drawing tools. OpenLayers allows map objects also to be observed by registering a listener to the map events (Langley, 2016.). OpenLayers' open source nature allows building applications on top of their ecosystem and it has drawing tools integrated to it. One of the disadvantages of OpenLayers is its limited interactions with the map's viewport.

Mapbox GL is an open source map library by a Mapbox company. Library enables developers to integrate map and navigation services to their applications. They provide solutions to both mobile and web platforms. Mapbox was founded in 2010 and in May 2013 they launched their vector tile rendering technology to replace traditional raster tiled maps. (Mapbox, About Mapbox, 2018.)

Mapbox's vector maps consume only one fourth of their traditional raster tile presentation which makes them fast and fluid. As described earlier vector map data enables storing contextual attributes to the features. These features can be queried in the client through Mapbox APIs thus reducing the need to make additional queries to a server. In addition to zooming and panning interactions Mapbox's maps can also be tilted and rotated. (Mapbox, Maps, 2018.) Thanks to Mapbox's efficient rendering, various interactions, open source nature and dynamic map styling it was chosen to be Steerpath's map library. Mapbox's open source ecosystem allows Steerpath to build their own SDK's on top of it thus making it easy to integrate to the final applications. Different interactions also provide new way of showcasing indoor maps and navigation. (Virkkilä, 2018.)

Summary of Map Library Comparison

The summary of map library comparison is presented in Table 1.

Table 1. Comparing different map library solutions.

	OpenLayers 3	**Google Maps API**	**Leaflet**	**Mapbox GL**
Map rotation and tilting interactions	partial			**X**
Drawing tools included	X		X	**X**
Support for own tile hosting	X		X	**X**
Vector presentation	X	X		**X**
Visual customisation of Map style	X	partial	X	**X**
Dynamic style changes on the client	partial			**X**
Support for plugins	X		X	**X**

As seen in the Table 1 Mapbox GL library has support for customising maps visually and adding plugins to the library thus making it the most suitable map library for the application. Another viable option would've been OpenLayers 3. One of the most evident advantages Mapbox GL had was that it some of the Steerpath's solutions were already built on top of it.

2.5 Steerpath Indoor Map Source Data

The source of Steerpath indoor maps are digitalised drawings of the customers buildings which are presented in CAD format. CADs can be existing digitalised drawings, or they can be drawn from the scratch by using relevant drawing software. Typically, there is one CAD file for each of floor of the building. While having separate CAD file for each of the building's floor keeps them organized it also makes map maintenance work harder. Should there be changes that effect whole building they need to be done to all of its CAD files.

Data exists in CAD drawings in different data layers. Steerpath's indoor maps are created by assigning essential data into designated CAD layers which are then processed through the Steerpath map server. This essential indoor map data can be physical or

abstract. Physical features include building outline, walls, floor layout and furniture whereas abstract features can include departments, rooms, POIs and wings. Abstract features may also be given tags and other contextual data attributes. A way to define how data should be presented in the final map is done by assigning attribute: “css_class” which is later interpreted by the map’s style object.

When indoor map data layers are selected from the CAD it will be uploaded to the Steerpath map processor. The result of the process is a GeoJSON file which will be stored in the database. Based on this GeoJSON file vector tiles are generated and encoded into mbtile file format. (Anttila, 2018.)

2.6 Steerpath Indoor Map Data Models

The extracted data from the generated GeoJSON are divided into four following categories:

- **Structures** are physical features of the building such as walls and floor. Structures don’t typically have contextual attributes assigned to them.
- **Visuals** are also physical features and they represent visual elements of the buildings such as furniture and stairs.
- **POI**s are more abstract features and they contain more contextual attributes such as tags, titles and identifiers.
- **Assets** include the beacon locations encoded as feature collection of GeoJSON points.

Steerpath’s APIs enables querying and writing to all these categories. (Anttila, 2018.) This thesis focuses on editing and updating the POI data from the data layers. At the most fundamental level the developed application can be described as a User Interface (UI) for communicating with company’s database and its indoor map data.

3 Application’s User Experience

As mentioned in the introduction, one of the main points of this thesis was to create a fluid and intuitive user experience for the application. The applications User Experience design started with analysing and reviewing different sort of existing map editors.

3.1 User Experience and User Interface

Terms **User Experience (UX)** and **User Interface (UI)** are crucial in software development, but they serve a different purpose in its design. User Interface can be thought as the space between the user and the product. User interfaces consist of inputs and outputs. Inputs are the interactions users have with the product and outputs are the feedbacks of their actions. Good user interfaces should allow products to be manipulated with minimal input in order to achieve desired output. Designing these interfaces where users' inputs generate desired output is called User Interface Design. (Moreno, 2014.)

User Experience is used describe persons feelings and behaviour towards the product. In addition to this it also includes how users perceive the products utility, efficiency and adaptability. User Experience Design is planning the experience users have with the product and finding out what emotions it creates in them. UX design aims to find out, what are the actions users would have with the product therefore it determines the UI design. In conclusion UI can thought as component of UX. (Moreno, 2014.)

Designing UX can include following matters (Xia, 2017.):

- personas
- user stories
- usability testing
- designing wireframes.

Personas are representations of types of customers. Personas are used to describe to whom the product is for (Bernstein 2015.). Typical persona to use application would be an employee of Steerpath who is making map updates for the customers. Persona would already have some understanding of the product which he/she is using and what it is capable of. A user requirement survey was conducted with the personas and its result are analysed in chapter 4. After testing the product internally and making sure it would meet the requirements the definition of persona could be expanded. A new persona could be a customer who is responsible for keeping company's maps up-to-date.

User stories are short and simple descriptions of the product's features in the perspective of its users. User stories help to embody the products features and what purpose they serve. Stories are typically written with the following template: "As a <type of user>, I

want <goal of the action> so that <a reason for the action>". (Pichel, 2016.) For example, for the map editor a typical user story could be: "As a customer I want to be able to add a single tag to all of my restaurants in my map data in order make them searchable".

Usability Testing is evaluating the product by testing it with its users. In usability test users are given tasks to complete with the product. During the test their actions and behaviour are being observed and written down. Usability tests are used to find potential problems with the product and measure the users' satisfaction with it. (Soegaard, 2018.) During the development process the application was tested from time to time with real map update tasks. The first official usability test was conducted when the product was launched internally. The usability test and its results are presented in chapter 7.

3.2 Analysing Existing Map Editors

The core of the UX design of the application was influenced by some of existing map editors. These map editors have established good procedures when it comes to editing geospatial data. Since many of the employees of Steerpath were already familiar with different type of map editors it was important that the new application follows the same principles and practises.

3.2.1 OpenStreetMap iD

From the existing editors most related to the new application is OpenStreepMap iD Editor (OpenStreetMap, 2018). ID editor is an online web-based editor which can be used to add and edit road and outdoor map data on OpenStreetMap ecosystem. The UI layout of iD consists of three main parts: Edit Feature Panel (1), Tools (2) and Map Panel (3) as seen in Image 1.

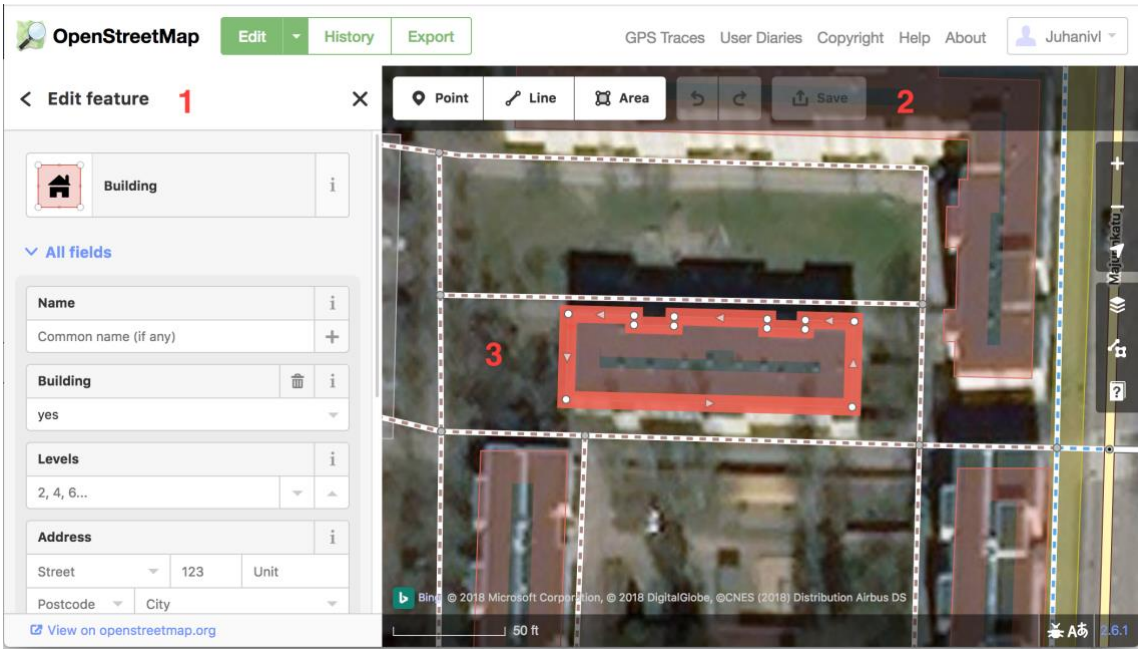


Image 1. The UI of OpenStreetMap iD Editor when a building is selected to be edited.

When the user either selects a feature from the map view or adds a new from the toolbar the sidebar activates. If the user has added a new object to the map a selection of different types of categories is viewed in the sidebar. Based on the selection of the category the sidebar inflates a form which allows editing feature's contextual data attributes. If an existing feature is selected from the map view the same edit form is shown automatically based on the contextual data. Components of this form are determined by the category of the object. For example, a restaurant will have opening hours selection in the sidebar whereas residential building wont. When the user is satisfied with the changes they can be uploaded to the server from the toolbar. (LearnOSM, 2016.)

3.2.2 DraftSight

Draftsight is a computer software which is used to edit 2D CAD files. When compared to OpenStreetMap iD editor DraftSight provides wider range of ways to edit objects' geometries. DraftSight UI layout can be composed of multiple different views. By default, DraftSight layout consists of four main elements:

1. **Properties sidebar** shows the properties of selected object and layer which it belongs to.
2. **Model view** which can be thought as the canvas where elements are drawn.

3. **Tool Matrix** provides tools for drawing the objects.
4. **Command Window** provides essential information about the state and mode of the editor and what objects user has selected.

These elements are shown in Image 2.

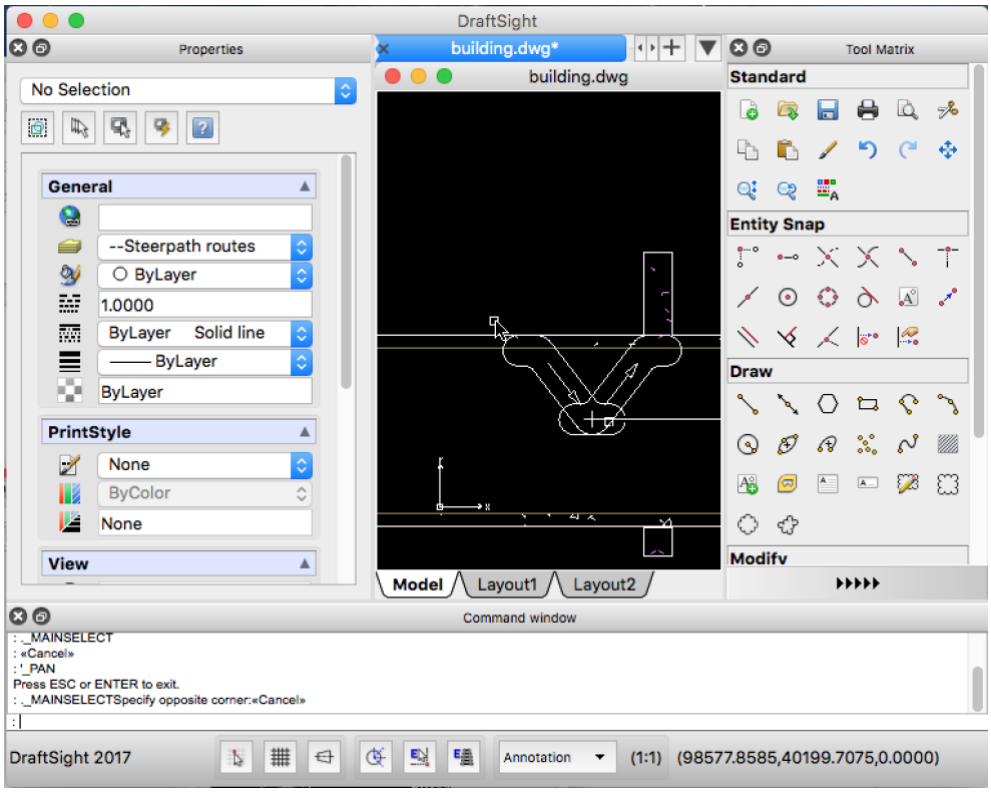


Image 2. Screenshot of DraftSight's layout which consist of four main elements: Properties, Model view, Tool Matrix and Command window.

While some of the DraftSight's features and drawing tools are relatively complex and advanced some of the concepts of selection and editing were thought and reviewed during the design of the application. The basic procedure of bringing a feature to edit is done by clicking it from the model view. The type of feature will determine the content of Properties sidebar much like with iD editor. The editing tools are listed on the right of the Model View and they are grouped based on their usage.

In addition to this DraftSight allows multiple objects to be selected and deselected in many ways. Objects can be brought to the selection with a box select, search or from a list of layers. The ability to select and edit multiple objects makes DraftSight an efficient tool.

3.2.3 Mapbox Studio and Maputnik

Mapbox Studio and Maputnik editor differ from iD and DraftSight since they are used to create styles for maps. Despite this, their UI layout follows the same principle as shown in Image 3 below.

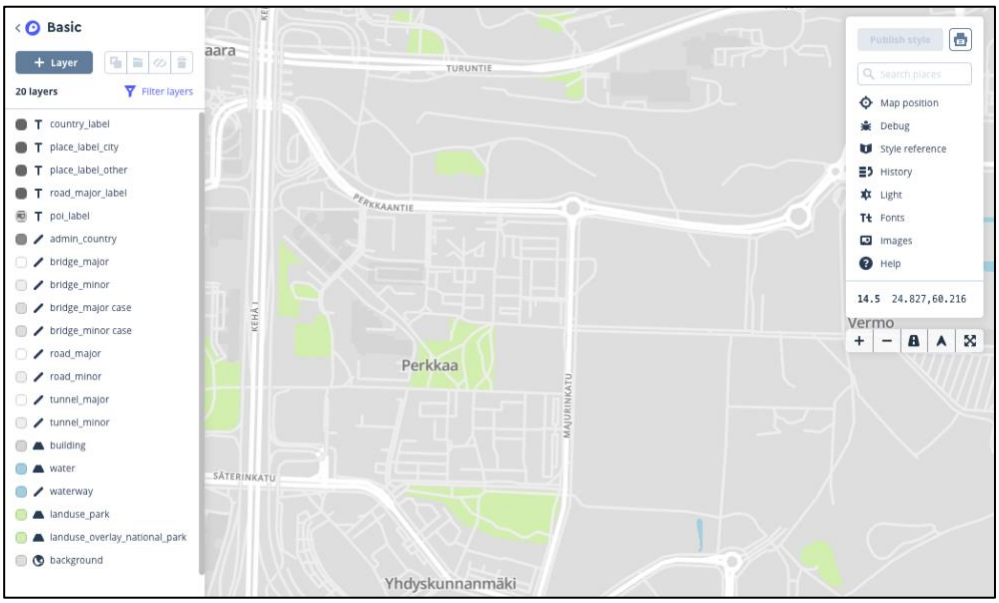


Image 3. Mapbox Studio's layout which consists of sidebar, floating info view and map view.

Both of these editors have map view and left sidebar to support it. Based on the element selected from the map the sidebar will update. Both Mapbox Studio and Maputnik emphasise on the size of the map view. The sidebar's elements are collapsible, and they will only require the needed space once they are activated. Since Maputnik and Mapbox Studio are used to create map style they need to show as much as the map data as possible.

3.2.4 Summary of the Analysis

From the use case and functionality point of view, most related to the application developed in this project is the OpenStreetMap iD editor. While iD editor focuses more on editing outdoor map data than indoor map data the concept of the editing process is the same. iD editor's UI layout consists of three simple parts which all react to the state of editing process. Editing existing objects is done by selecting them from the map and new ones are created from the sidebar.

Compared to iD editor's DraftSight offers more complex yet powerful ways for editing object's geographical data. The selection can be done multiple ways and how DraftSight supports multiple selection makes it and efficient tool.

One thing that all these four benchmarked editors have in common is that the effects of the changes done in their sidebars can be seen in the map view instantly. When the map view responds to the changes done in the sidebar immediately it makes the editing process more intuitive. The immediate responses of the edits were one of the requirements which was raised in the user requirements survey.

4 Gathering and Analysing User Requirements

In order to get the requirements for the application, a survey was conducted among four employees of Steerpath. The survey was designed for the employees who were actively working with the map production and maintenance, i.e. personas. The survey was conducted at the company's office and it took approximately 20 minutes from each of the employees. There were four employees who took part in the survey.

The survey had four questions regarding the current state of the map maintenance and requirements for the application. The whole survey questionnaire can be found in Appendix 1. The answers set the minimum requirements for the application and gave the direction for the development work. The questions were following:

1. What are the most common map changes?
2. What are the required procedures in order to make these changes happen?
3. Are there some good features or practises with any of the existing tools/editors (i.e. DraftSight, OpenStreetMap Editor, Maputnik editor or Mapbox Studio) that you would like to have in the new editor?
4. Please paste here copy of your most recent DraftSight commands. You can do this by right clicking Command Window and selecting "copy history".

After receiving the answers from the employees, they were analysed and summarised by grouping the answers by their similarities. The answers can be found from the Appendix 2. First question was about the most common map updates customers want to be done to the maps.

4.1 Question 1: Most common map updates

The answers for the first question varied greatly, below is a summary of how the answers are grouped into coherent findings.

Editing the Attributes of the Data

Most common change customers wanted to be done to their maps and its annotations was to edit existing POIs. These edits mean changing some of the properties POI object such as title and class. There were also comments on modifying the area of POI but not so much about changing their original location. Second and third most requested changes were adding and removing POIs from the map. Adding a POI representing that an area is under maintenance or renovation was specified as one of the use cases. According to the employees, these temporary changes are common because customers want to have their maps up-to-date.

Changing the Visual look of the Map

Sometimes customers want to change the visual look of their maps. Changing the visual look of the maps means that visual outlook of POIs should be editable such as area and icon colour. According to the employees updating the visual data layer is also requested by the customers. This means that walls, furniture, floor plan and building outline should be editable. Objects in visual layer do not have properties or attributes like POIs thus updating them would emphasis more on editing their geometries in a convenient way. Since the visual features are stored in the vector tiles and not in GeoJSON editing them would first require map editor to request visual features from server. Adding additional visual features to the map for example furniture would require editor to have a collection of predefined GeoJSON shapes to be in stored.

Changing the Positioning Data

Updating assets such as beacons was also requested by the customers. Updating beacon data has a straight effect on the positioning system. Editing beacon data is relatively more straight forward compared to POI editing since they have less properties to be edited. These properties are:

- RSSI which can be used to adjust beacon signal strength and how it will be received by the position algorithm.
- Major value which can be used to categorise beacons for example by their colour.
- Minor value which is used to reference a single beacon.

In addition to these properties beacon locations tend to change relatively often which means that the new map editor should have a way to move these beacons on the map. Other position and navigation related map changes are route updates. These updates are usually tied in with changes inside venue that will prevent routing through certain areas. In order to update existing route network in the web Steerpath would have to make changes to their routing algorithm. Therefore, the priority of the project was with the map's POI objects and not in the routes or assets.

4.2 Question 2: Current Map Maintenance Procedure

The second question was about the current map maintenance procedure and what it requires to make map updates to happen. One employee described the update process to be following:

- After receiving a ticket for a map update it has to be logged into an issue tracker worksheet.
- Small changes can be done with a REST API call to the server, for example with a Postman app, but more complex changes need to be done to the source data, CAD drawings.
- After finding the customer's right CAD file and making the necessary changes to them they need to be republished.
- During republish the Steerpath server will parse the CADs and generate indoor map and route tiles. Since each floor is in its own CAD file one building may consist of multiple files. According to the employees republishing CADs may take from couple of minutes up to one hour.

One employee stated that not being able to see if changes that have been made have worked until the CADs have been republished is one of the disadvantages of the current system. Also, not knowing how style ruling works makes it hard to understand how changes that have been made to the CAD are connected to the final presentation of the map. Employees universally indicated that making the feedback loop quicker would likely reduce mistakes, speed up maintenance and improve usability greatly.

4.3 Question 3: Identifying Key Features in Existing Solutions

In the third question the aim was to find out if there were some good features or practises in any of the map related editors or tools that the employees of Steerpath had used and wanted to see in the new application. Even though some of the existing editors or tools mentioned as an example are used edit different sort of geometrical data they still have well thought practises that could prove beneficial. For example, Mapbox's Mapbox Studio is all about creating and editing the style document and not about editing the data itself. The examples of existing editors and tools mentioned in the question were:

- CAD editor DraftSight
- online map editor OpenStreetMap iD
- online open source style editor Maputnik
- Mapbox's style editor Mapbox Studio.

Having familiar features and practises from these already used tools would make the new editor easier to adapt to.

Based on the answers received from the question, the employees of Steerpath wanted the new application to have practises that would make it an efficient and a powerful tool capable of making multiple edits simultaneously. There were also mentions about obvious features which any web application, and especially editor, should have. These included keyboard commands and visual confirmations of the actions. There were also examples of features that would prove useful for more advanced users.

The employees mentioned three examples of how multiple selection was solved in the other editors. One example was being able to draw a rectangle to the map and selecting all the features inside it was seen as one way of doing this. Another way was first selecting one object and then selecting similar objects based on given category. There was also mention about being able to search by given category and then selecting the matching results. After the objects were selected the editor should be able to edit all of them simultaneously.

One employee mentioned a use case where a single tag needed to be added to all of the POIs in the map. Being able to select multiple objects for the edit and editing them all simultaneously was the most appreciated and wanted feature which was mentioned

by 3 out of 4 employees in the survey. Having this feature in the editor would make it a powerful and efficient tool.

According to the employees, the application should also have the most common keyboard shortcut commands like any other application. These include undo/redo, copy/paste, confirm/reset and select/deselect. Based on feedback triggering actions from these universally familiar commands would make the application quick and easy to use. Some of these shortcut commands are controlled by the keyboard keys and some are combinations of certain keys and mouse click events. While some of the keyboard commands are straight forward the fact that Mapbox's map already uses some of the keys in its native interactions will make them hard to implement.

More advanced users who have better understanding the concepts of GeoJSON, the division between the data and styling in Mapbox and how Steerpath indoor map data models work want to have more advanced features. Some of the employees mentioned that they appreciated how Maputnik and Mapbox Studio support editing directly the style JSON object in a text area. Seeing the objects which are being edited in a JSON format could also help advanced users to validate and verify the data. Another employee mentioned Mapbox Studio's feature to show all the map's source data in a translucent way helps to get deeper understanding of the data.

4.4 Question 4: List of Most Common DraftSight commands

As stated in chapter 2 Steerpath's maps source data is encoded in CAD format. Editing the CAD files requires a CAD editing software. Most commonly used software among the Steerpath's employees is an editor called DraftSight which was also analysed User Experience-wise in chapter 3. Since DraftSight is powerful and efficient tool for editing geographical objects understanding and possibly following its editing practises in the application would be beneficial.

One way to analyse the DraftSight's editing procedures is to research its users' commands. DraftSight allows to print the latest commands it has received from the users. In the last question of user requirements survey employees were asked to copy these commands into the survey. The commands and states received from the employees were tabulated and analysed and they were used in the design of the application. The

summary of the most repetitive relevant commands and states are presented and described in Table 2. The total amount of relevant commands was 4643. The whole table of DraftSight commands is presented in Appendix 3.

Table 2. Commands with the most frequency and their description

Command	**Description**	**Frequency**
Specify next vertex	When drawing a polygon specify where next vertex to be drawn	882
MAINSELECTSpecify opposite corner:	When using box selection specify the opposite corner of to which the box will be drawn	733
Options: Arc, Close, Halfwidth, Length, Undo, Width, Enter to exit or	When object is selected print editing options	729
MAINSELECT	Select object for editing	636
Cancel	Cancel editing process	313
Options: Enter to continue from last point or	When drawing inform the user about possible options	233
Specify start point»	When drawing starts specify the starting point	209
All layers have been turned on.	When switching layer visibility inform the user the state	147
Undo SHOWLAYERS	Undo of SHOWLAYERS command	144
Undo INTELLIZOOM	Undo of INTELLIZOOM command	128
POLYLINE	Draw a polyline on to the canvas	110
Specify destination»	Specify destination of draw	94
STRETCH	Polygon objects can be stretched by selecting their middle vertex and dragging it in the canvas	46
Stretch point»	Corners of polygons or lines can be stretched by selecting a single corner vertex and dragging it in the canvas	45
Specify next vertex» «Cancel»	Cancel drawing polygon or line	31
Undo DELETE	Undo DELETE command	26
Undo Modify Property	Undo attribute edit	24
Specify next vertex» _Undo	Undo specifying next vertex	22
: POLYLINE	Select polyline drawing tool	16

Auto save	Auto save edits	16
Redo INTELLIZOOM	Redo INTELLIZOOM command	11
: _DELETE	Delete selected objects	10
Undo SNAPGRIP	Change the drawing mode of the editor.	6
qselect	Select objects by given properties	6
SMARTSELECT	Select objects by filtering their properties	6
PASTE	Paste object to the canvas	5
Specify entities	Specify entities for edit selection	5
Already zoomed in as far as possible.	Inform the user that current view port is zoomed as far as possible	5
GROUP	Group selected objects into one	5

As seen from the table the most common interactions users have with the DraftSight are related to selection and deselection. Since DraftSight is mainly a drawing tool lot of the most common commands after selecting and deselecting are related to drawing. These drawing commands include specifying next point or vertex, stretching existing shape and selecting a starting point. Commands such as undo, delete, paste and save are evident commands in every software and they were used relatively often with the DraftSight as well. Having an ability to undo and redo actions gives users ability to fix mistakes they may have done with the editor.

Commands received from the survey helped to map the most fundamental interactions users typically have with editors. From analysing the DraftSight commands, the following were identified as key features for the application:

- Selecting and deselecting objects for editing
- Drawing interactions such as specifying next vertices and stretching
- Undoing and redoing interactions.

These interactions and practises users had with the DraftSight would be emphasised in the application as well. Most common commands also confirmed other findings found from the user requirements survey such as how selection can be accomplished many ways.

4.5 Summary of User Requirements

After analysing the answers of the survey, they were summarised into a list of functional requirements. These requirements, their ids, descriptions and priorities are presented in Table 3.

Table 3. List of requirements, their descriptions and priority levels.

ID	Requirement	Description	Priority	Source
Req1	Edit POI attributes	Edit POI's attributes such as title, identifier, class and tags.	High priority	Question 1
Req2	Change geometry of POI	Move POI's representative point and add or remove its corresponding area. Area should also be editable.	High priority	Question 1
Req3	Add and remove POI	Add a new POI the map or delete an existing one.	High priority	Question 1
Req4	Edit Beacons	Edit beacon attributes and change beacon location.	Medium priority	Question 1
Req5	Change routing	Edit routing network and disable routing through certain areas.	Low priority	Question 1
Req6	Customize POI's visual appearance	Give POI visual attributes like fill colour and icon-image and make it match to an override styling.	Medium priority	Question 1
Req7	Edit and add visual data	Edit walls and floor, walls and furniture polygons.	Medium priority	Question 1
Req8	Instant feedback for actions	When editing a feature show the results real-time in the map view.	High priority	Question 2
Req9	Edit multiple POIs simultaneously	Ability to select multiple POIs for the editing and applying changes to all selected POIs simultaneously.	High Priority	Question 3
Req10	Shortcut commands	Perform common actions from keyboard shortcut commands.	Medium priority	Questions 3 and 4
Req11	Show and edit data in JSON format	Show the raw data in JSON format and provide way to edit JSON directly	Medium priority	Question 3
Req12	Give information about current map view and data.	View map data in a translucent way so that objects that all map data will be visible.	Medium priority	Question 3

After analysing the most requested map updates and summarising the requirements for the application the planning of the implementation started. Since the most necessary and wanted features were related to be editing, adding and removing map objects such as POIs the application that developed at first was Indoor Map Editor.

While some of the user's answers referred editing different parts of the Steerpath's map data the application presented in this thesis focuses on the Indoor Map Editor. This is done in order keep the scope of the thesis manageable and the research in-depth. While editing POI data differs from editing position related data such as routes and beacons some of the editing practises and procedures could be implemented to the other editors as well in the future. Steerpath wants the map data to be editable from single dashboard the Indoor Map Editor will work as a prototype for the future editors.

5 Application's Architecture

One of the biggest disadvantages of the current map maintenance system is how long it takes to see changes that are done to the source data go live. A faster feedback loop was one of the requirements that was dissected from the requirements survey. In order to provide faster feedback from the user's actions a platform for the application was chosen to be web based. Having the application running on a web browser would make it accessible and easier to adapt to. The application's architecture is described and presented in this chapter.

5.1 Data Flow of the Application

The data that will be edited in the application can be divided into attributes and geometry. When any of the attributes or geometries changes a state manager that will sync the edited attributes on to the selected POIs. One of the requirements of the application was to show changes that are done to the objects in real-time (Req8). Providing faster feedback loop is done by state manager calling a map update after data synchronization. After completing the changes, they can be uploaded to the server. After the upload process is completed new map tiles can regenerated. The data flow of the application can be described as the communication between the state manager, map updater,

upload processor, attribute editor and geographical data editor. The communication is presented in the Image 4.

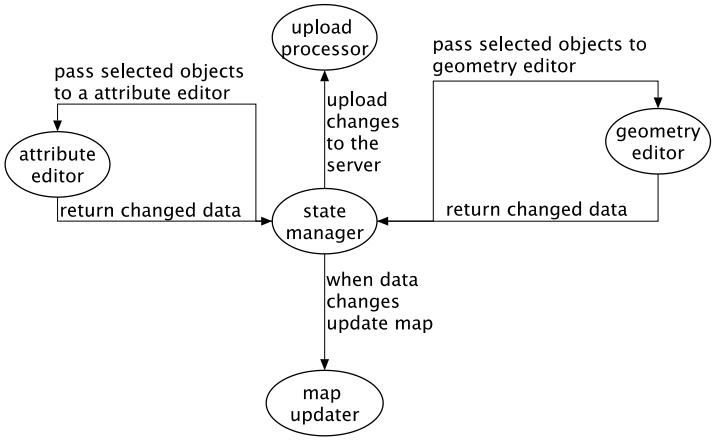


Image 4. Data flow of the application and most fundamental parts it consists of.

Since many of the attributes of the map objects are encoded as strings inside the GeoJSON's properties object the first approach for editing them would be from a web form. The form would consist of text inputs and select components. For editing the geometrical data most intuitive way would be to edit the geometries directly from the map view. This would require additional drawing tools integrated on to the map and a way to bind the updated data onto the selected objects' geometries. How Mapbox and its drawing tools can be utilized is described later in this chapter. Uploading edited data to the server will be done in utilizing standard HTTP request protocols in web.

5.2 Mapbox as the Editor Component

Based on Chapter 2, Mapbox GL library was select as foundation of the Indoor Map Editor for viewing and managing the map data. Mapbox was chosen because it's an open source library and it allows adding plugins to the map. Plugins are required in order to show indoor map data and edit geometric data with appropriate tools. It also has API's which provide many ways of interacting with the map data and how it should be

presented on the web. APIs allow the map view to be queried from data and making changes to the map's style object which can be seen as dynamic updates. Based on the user's interactions the API fires events which can be registered to be listened to. In order to utilize Mapbox's ability to showcase interactive and dynamic maps in the Indoor Map Editor it is important to understand how the map data is bound to the style and what API methods should be used to make the best of it.

According to Mapbox style specification the style document determines what objects will be shown in the map, in what order and how they are drawn. Creating a style for maps can be thought as writing rules how objects should be presented. In order to make the map source data appear style object needs to have layers referring to them. One data set, also known as source, can have multiple layers referring to it thus enabling styling it many ways. (Mapbox, Style Specification, 2018.)

Relevant data can be filtered out from the source based on its objects' properties by writing filter rules to the layers. For example, a polygon might have a property that it's an ocean. Respectively in a style layer's filter can be stated that all polygons with property "ocean" should be drawn as blue to the map. This same principle about filtering out data can be applied to indoor map objects. Points representing POI in the building can have properties stating that which class of POI they are and to which building and floor they belong to. By adding a filter to match certain type, building and floor the map will show corresponding data. In Listing 2 there's a layer which will show only those points whose "css_class", "buildingRef" and "layerIndex" properties match its filter's rules.

```
{
      "id": "cafe-pois",
      "type": "symbol",
      "source": "blueprint",
      "source-layer": "blueprint",
      "minzoom": 16,
      "filter":
            ["all",
            ["==", "$type", "Point"],
            ["==", "layerIndex", 2],
            ["==", "buildingRef", "67"],
            ["==", "css_class", "café_poi"]
      ],
      "layout": {
            "icon-image": "{icon}",
            "text-field": "{title}",
      },
      "paint": {
            "text-color": "#000",
      }

}
```

Listing 2. Layer which will show all the POIs from blueprint source that match the filter's css_class, layerIndex and buildingRef values.

If the "css_class" filter in Listing 2 is changed a different set of POI objects will be shown and the map view will update. In practise this can be seen as icons changing in the map. One of the user requirements from the survey was to have an instant feedback of the changes that are being made to the POI. The detailed implementation of this requirement is covered later in the implementation chapter.

Mapbox's Plugins

Mapbox GL Draw is a plugin created by the Mapbox team which allows GeoJSON points, lines and polygons to be drawn on top of the map. Draw keeps track on the edited geometries and fires various events based on its state. Draw also has different types of modes which are determined by the state of the selection. When a GeoJSON Feature is passed to Draw its mode changes and the geometry will be highlighted in the map. Selected GeoJSON lines and polygons will have vertices drawn in their corners and in the middle of their lines. By dragging these vertices, the geometries will change. In the Indoor Map Editor by clicking a POI and bringing it to the selection will also pass its geometric data to Draw instance. Geometric data can be the POI's representative point and related polygon area.

Steerpath Web SDK is a library which enables utilization of Steerpath indoor maps on top of Mapbox GL JavaScript library. The SDK keeps track of the buildings that user is viewing, and fires events based on interactions. On each map movement, the SDK will query the map view for Steerpath building data and store it. If there is a building in the view, the SDK will make it active by activating a floor switcher and selecting either the latest floor index or default layer index to be shown. When a building becomes active a floor switcher control panel will be created for the map. The SDK is needed in the Indoor Map Editor in order to show the indoor maps and floor switcher but also when creating a new POI on the map the editor needs to know to which building and floor it will belong to. This building data can be accessed through the Steerpath Web SDK's APIs.

5.3 Backend Communication

The POIs in Steerpath data are encoded as collection GeoJSON points in the server and they can be queried and written with Representational State Transfer (REST) API methods such as GET, PUT, POST and DELETE. Since the Steerpath's POI data is already encoded as GeoJSON the Indoor Map Editor would also use it as data format. Backend queries can be focused by giving URL parameters to the queries. For instance, server will return a single POI Feature when it is given its id as URL parameter. Backend communication is needed in the Indoor Map Editor when:

- User signs in to the application with an API key received from the company the application will query server for map data with the given API key. If there's map data related to the key the login will happen.
- User wants to select a POI from the map by clicking it and the POI data is queried from the backend.
- User is ready to publish changes that are done, the application will update the backend. POST method will add new created POIs, PUT will update existing ones and DELETE will remove the deleted ones.

As stated in chapter 3 at the most fundamental level the Indoor Map Editor can be described as an UI for the backbend's requests.

5.4 UI Design of the Application

Wireframe created for the application can be seen in Image 5. It was made to follow familiar guidelines from other editors which Steerpath employees had used. As seen in the image the layout of the application consists of four main elements: (1) **Toolbar**, (2) **Sidebar**, (3) **Map view** and (4) **Bottom bar**. The whole edit process can be managed through these four elements.

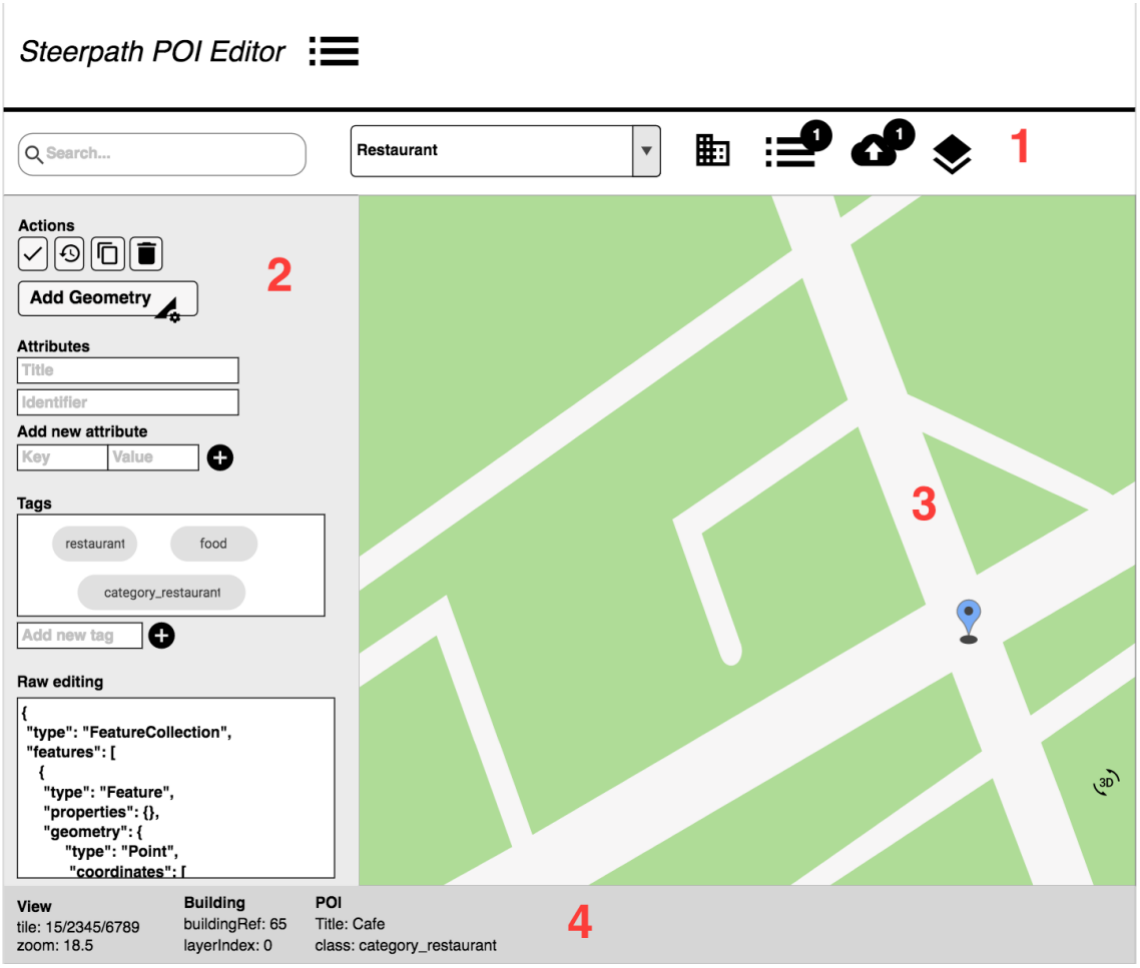


Image 5. Application wireframe presenting state when a POI is selected. The blue marker in the image represents a POI and the sidebar is populated by its data.

Sidebar

The Sidebar (2) activates only if there are POIs in the edit selection. The sidebar's role is to show and edit contextual attributes of selected POIs and handle the state of the edit process. The sidebar can roughly be divided into action section and form components. From the action section users can manage the state of edit process and from the form components change the attributes of POIs. Form components are text inputs, tag selector, colour picker and JSON text area. Each form component changes one attribute of the POI.

The first elements to appear in the Sidebar component are action buttons which can be used to change POI's class, confirm current edit, restore the POI's state, delete the POI, select similar classes of POIs and add or remove area of POI. Since all of these actions

are only needed when there's POIs in the selection they're embedded in to the sidebar. After the action section there's form components.

Text inputs are used to change POIs title, identifier, description and searchable keywords. Title and identifier are shorter single line values, but description and keywords require multiple line inputs. Users can also add a new text input component and sign it to a new attribute and value.

Chip selector is used to select and deselect tags for the POIs. If there are more than one POI selected and there exists tags that are only partially selected, they are presented as light blue whereas tags that are selected by all the POIs in selection are highlighted as dark blue. In the JSON text area the whole FeatureCollection of selected POIs can be edited. The raw presentation of the data also gives more advanced users better understanding of the POIs in selection

Map view

Sidebar is tightly coupled to the Map view (3). For example, changing the title of a POI makes it also change in the Map view. While the sidebar allows editing a POI's contextual attributes, the Map view allows editing its geometric data. A POI's point can be dragged, and areas can be modified and stretched. The Map view has Steerpath's floor switcher controller which allows switching between floors of the building.

Bottom bar

Underneath the sidebar and Map view there's Bottom bar (4) which gives users information about current state of the editor. Current map view position, selected building and possible POI object underneath the mouse pointer are shown in the Bottom bar. As the view changes in the map the information in the bottom bar updates. Bottom bar informs the user when there's editable POIs in under the mouse pointer.

Toolbar

The Toolbar (1) has five buttons and a search bar. Each of the button of the Toolbar can be pressed regardless of the state of the editor. From the Toolbar users can view source in a transparent, open the list of edited features or buildings, upload changes and

generate tiles. In both buttons list of edited features and upload edited features there is a badge displaying the number representing the amount of POIs in both cases.

The edit process can be divided roughly into following steps:

- User hovers around map view and POI under mouse pointer is shown in the bottom bar indicating that it can be edited.
- User either selects new POI from the dropdown list or an existing one from the map by clicking it.
- The sidebar is populated by the action section and form components which provide way to change the POI's contextual attributes. In the map view the geometries of POIs in selection are highlighted.
- User changes attributes of the POI and changes can be seen in the map.
- User confirms edits and number near the toolbars upload button changes indicating that there's one change to be uploaded.
- User uploads changes from the toolbar and application prompts about the result of the upload.

In the next chapter the implementation of the requirements for the application is described based on the designs and wireframe.

5.5 Selected Tools and Development Environment

Development environment of Indoor Map Editor consists of tools and solutions that are commonly used in web application development. For managing all the required libraries and packages Node Package Manager (npm) was chosen. Npm is the standard package manager for the Node.js which is a runtime environment for web applications. When installing Node.js, npm will included as a recommended feature. Npm consist of command-line client interface, cli, and database of packages. This database is also referred to as npm registry and it can be accessed either from the cli or from the npm website (Wanyoike, 2017.). Packages downloaded from the registry can be thought as a building blocks for the project and they can either be:

- node modules used in the server side
- command line tools
- packages that can be implemented into front end web applications (npm, 2018).

While npm handles the dependencies between the packages used in the project webpack, a Node-based tool, handles the module dependencies inside the project. Webpack is a powerful tool which can be used to bundle modern written modules into common JavaScript and static asset files. Webpack provides multiple ways to configure its usage inside the application. Entry point determines the point from where webpack starts to build dependency graph and output is the destination where the bundled projected will be created to. By default, webpack can only understand JavaScript but having loaders enables it to process different file types as well and adding them as modules. Loaders have to principles which way they are constructed. Property “test” determines which file types webpack should take into consideration and “use” defines which loader should handle that type of files. To expand to the usability of webpack its configuration can be given plugins to perform various tasks from bundle optimization to environment definition. (Webpack, 2018.)

In order to utilize ECMAScript, the latest version of JavaScript syntax, a compiler called Babel was used. While all the browsers don’t support latest features in the language Babel helps to compile the language to a supported version. In the release of ECMAScript 2015 a concept of classes was introduced. Classes provide a way writing cleaner prototype-based object-oriented patterns in JavaScript. Exporting and importing classes enables writing modular and reusable components which will make the architecture of the application simpler. (Scerri, 2017.)

Class syntax can also be used with React library which was used to create user interfaces for the application. React is a library released in 2013 by Facebook and it allows applications to manipulate the DOM based on the state of the components. With React rendering only the part of the DOM that requires rendering makes applications more fluid. (Gackenheimer, 2015.)

6 Implementation of the Requirements

The implementation of each of the user requirements is described in this chapter. The implementation work started from the most common and required use case and progressed towards requirements with less priority. Requirements are referenced with the same identifiers as in Table 3.

6.1 Component Structure

The application is divided into components. Components allow creating small individual and reusable building blocks which have dependencies between each other. While the application allows routing between the views the most fundamental component is the editor component and its child components. The data flow between the editor component and its child components is presented in Figure 1.

As seen in the Figure 1 the parent component which is called Editor, has three child components: Toolbar, Sidebar and MapComponent. Some of the child components might also have smaller individual components from which they consist of. For example, Sidebar consists of multiple form components which each handle editing POI's contextual attributes.

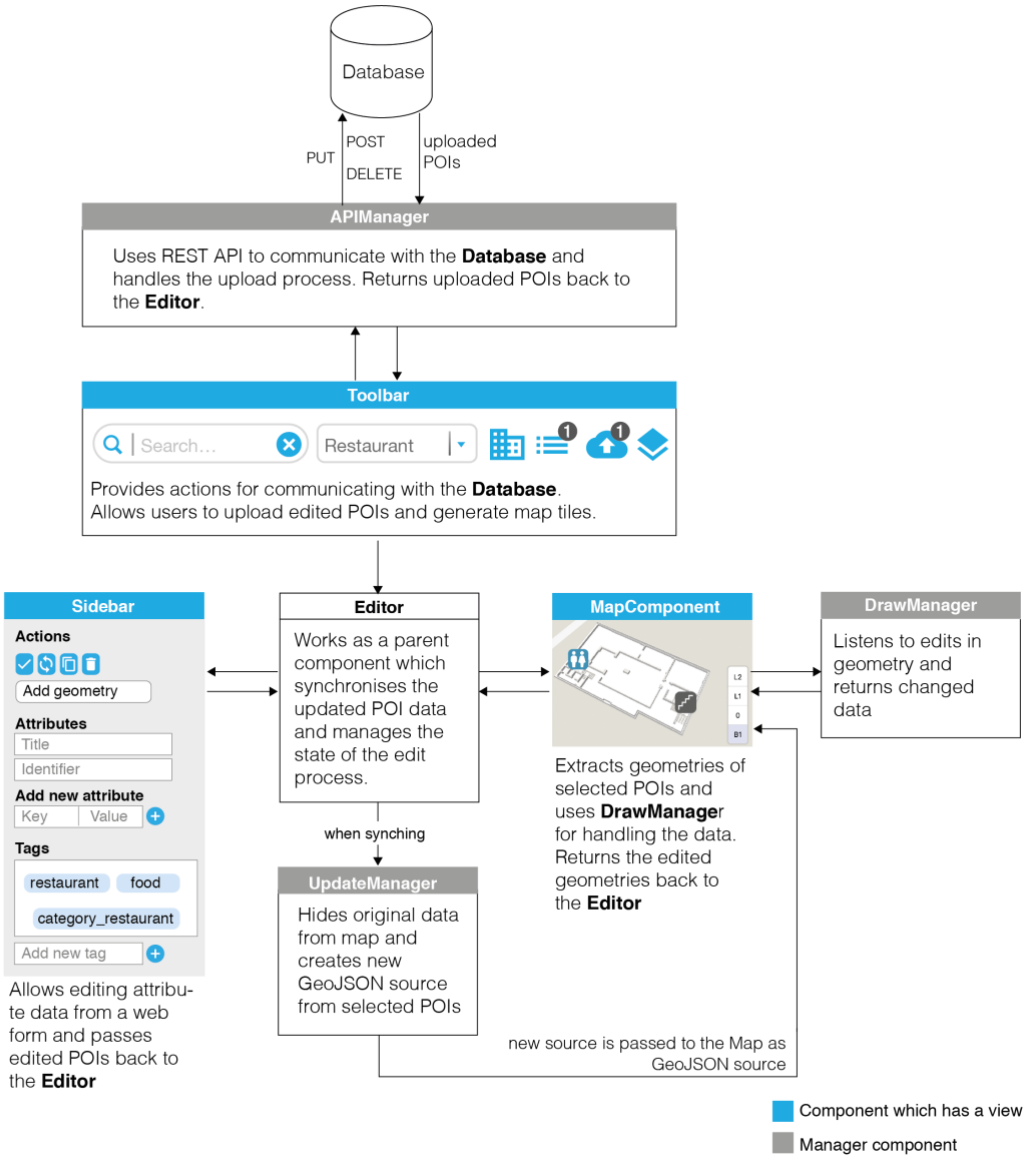


Figure 1. Data flow diagram of the Indoor Map Editor.

The way editing works is that each child component edits a single attribute or geometry of the POI collection in selection and passes the edited POI collection back to the Editor component which is located in the middle of the Figure 1. When receiving an updated POI collection, the Editor component synchronises and updates its state. When the state updates in the Editor component it is passed on back to the child components. Therefore, each child components of application will have the latest version of selected POI

collection. This data binding can be seen in practise when title property of the POI is changed in the Sidebar it is also updated in the MapComponent.

In addition to the parent and child components the application has managers which are presented as grey in Figure 1. Managers can act as a collection of static helper functions or as more versatile classes.

The application has the following managers:

- APIManager handles the server requests and returns their responses back to the component. Since requests to server are asynchronous many of the functions of APIManager works as promises.
- DrawManager handles the geometric data that is passed from the selected POIs. DrawManager also listens to the events that happen when these geometries are changed by the user and returns them to the Map component.
- GeometryUtils is a helper class which is used to perform geometric calculations for example when area is added to a POI GeometryUtils calculates its initial coordinates.
- UpdateManager keeps track of the edited POIs and hides them from the original source. UpdateManager allows showing changes in real time.
- KeyboardManager can be used to bind certain actions to a keyboard commands.
- NotificationManager is used then the application needs to notify the user of something. NotificationManager can create a dialog, popup or loading screen.

When there are actions and data management which are needed in multiple components it is more practical to have reusable helper functions and managers to perform them.

6.2 Basic Editing Procedure

The basic editing procedure can be divided into four steps which are presented in this chapter. The procedure starts with either selecting an existing or creating a new POI to the map. Selected and edited objects can be uploaded to the database.

6.2.1 Selecting and Deselecting POIs

Selecting and deselecting is one of the core features of the Indoor Map Editor and they were also listed as some of the most common DraftSight commands in the user requirements survey. When the user clicks a POI object in the map it is brought to the selection through a couple of steps. During the implementation process, it was discovered that even the simple operation of selecting a Feature can be fairly complicated.

While the Mapbox API provides a way to query map viewport from rendered objects, the method queryRenderedFeatures has a couple of impediments. Since some of the objects may exists in two different vector tiles their geometries may be cropped by tile boundaries and geometry simplification. In addition to this method queryRenderedFeatures doesn't return objects' ids. Therefore, POIs identifiers are stored in the object's properties object. The limitations of queryRenderedFeatures method means that in order to get the whole POI object from the click event it needs to be queried from the Steerpath database by its id value. When server returns a POI object for the selection it's added to a list of edited POIs. In Image 6 there's a screen capture of the Indoor Map Editor when a POI is selected from the map.

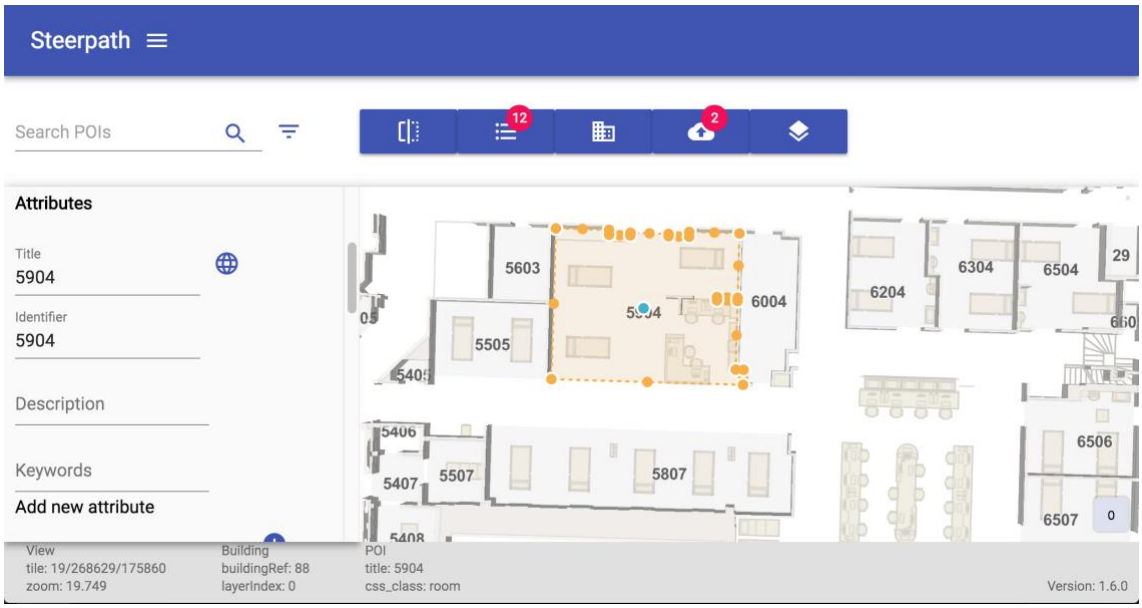


Image 6. POI representing a room 5904 is selected.

Based on the POI's class the editor's sidebar is populated by appropriate form components as seen in the Image 6.

6.2.2 Adding a new POI

Adding a new POI to the map follows a slightly different principle than selecting an existing one from the map. The first step of adding a new POI to the map is done by selecting its class i.e. printer, toilet or restaurant. In the Steerpath data the class is defined in the properties as “css_class”. Selection of POIs can be done from the sidebar’s dropdown menu. The dropdown menu consists of all the possible classes of POIs that can be added to the map and each of them has a unique value assigned to it. When the class is selected the POI is given a temporary identifier by the editor. Having a different type of identifiers for edited existing POIs and new added POIs helps the application to distinguish them. This separation is also used in the upload process. By default, new added POIs are positioned at the centre of the map view.

6.2.3 Editing

POIs can be edited either from the sidebar component by changing their attributes (Req1) or from the map component by changing their geometry (Req2). Any changes that are done will trigger an update for both of the components. For example, writing a new title to the POI passes the information from the sidebar component’s text input field to the parent component which updates also the map component. This example is presented in Image 7 where the title of the selected POI is changed.

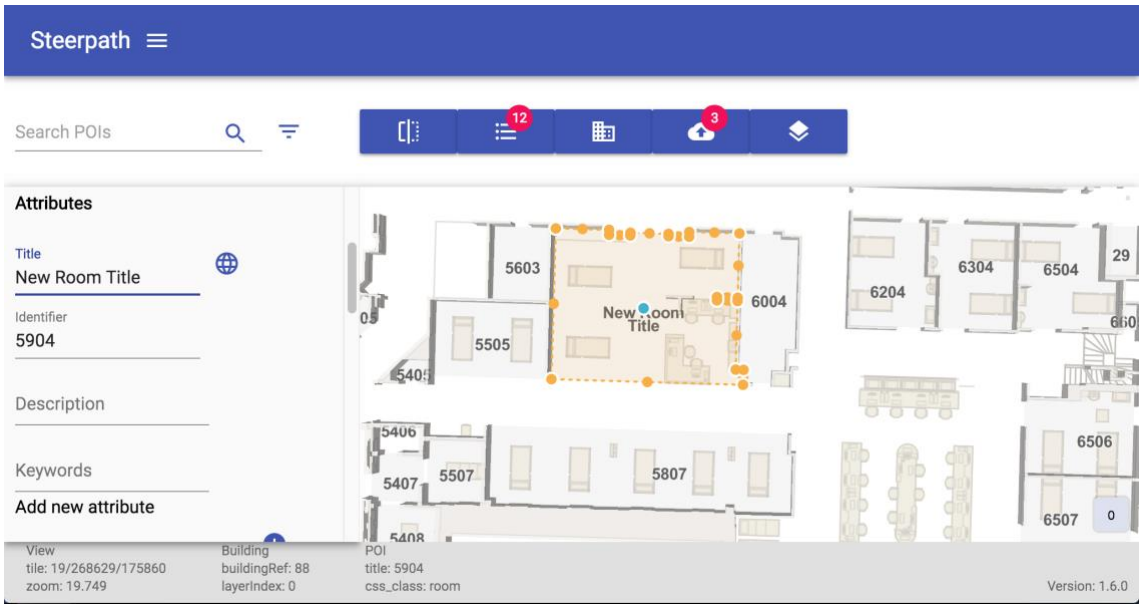


Image 7. When title is changed in the sidebar it is updated in the map view as well.

When the POI is edited in any way its property “saved” is set to be false. This is a way to distinguish which POI’s that have been selected should be uploaded to the server with database when the upload process starts. When the user has done changes to the POI they need to be confirmed. Confirming the changes can be done in three ways: clicking a confirmation button from the sidebar, clicking anywhere from the map or pressing enter key from the keyboard. If the user deletes a POI (Req3), it will be hidden from the map and its property “deleted” is set to be true. When the map component receives the collection of edited POIs it updates the source of which features should be shown by their “deleted” property.

6.2.4 Uploading

When the user is ready to upload the changes to the server the list of edited POIs is passed to the APIManager. The APIManager separates the list of edited POIs by grouping all the new, existing and deleted features in their own lists. The separation is presented in Figure 2.

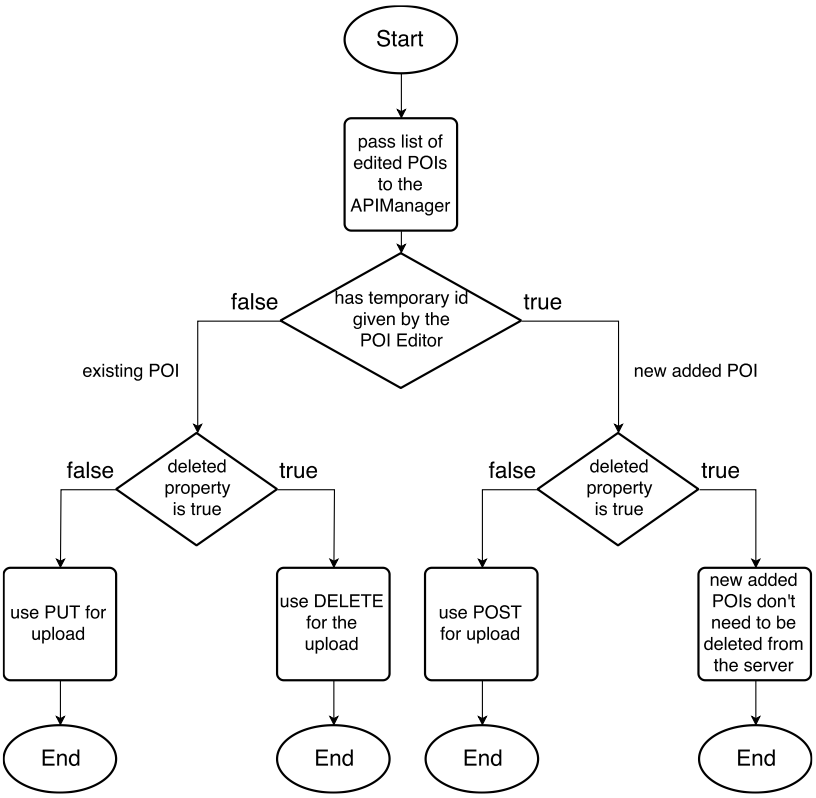


Figure 2. APIManager’s separation logic for the upload process.

The changes are uploaded to the server by using REST API methods. All the new POIs are uploaded by using POST method. Edited existing POIs are updated to the server with a PUT method and deleted ones are removed with a DELETE method. While the upload process is happening each of the POIs “saved” property is set to be true. When the upload is completed the processor returns the list of edited POIs in case for the further edits. After the upload process users can regenerate their map tiles from the updated GeoJSON data.

6.3 Multiple Selection, Deselection and Edit.

One of the most requested features from the user requirement survey was to be able to edit multiple objects simultaneously (Req9). After finding out of how the procedure of single selection, edit and confirmation should work the procedure was expanded to support multiple editing. In multiple edit each of the changes that are done will be applied to all POIs in the selection. Selecting multiple POIs for the edit can be accomplished couple of ways:

A standard way of creating a multiple selection and deselection in software development is by combining the mouse click event with additional keyboard command. In some applications holding either a control or alt key while clicking objects toggles their selection and allows multiple selection. By following this principle, the map editor allows multiple objects to be selected and deselected from the map by clicking them while holding the Alt key. The logic of how multiple selection works with combined alt and click event is presented in Figure 3.

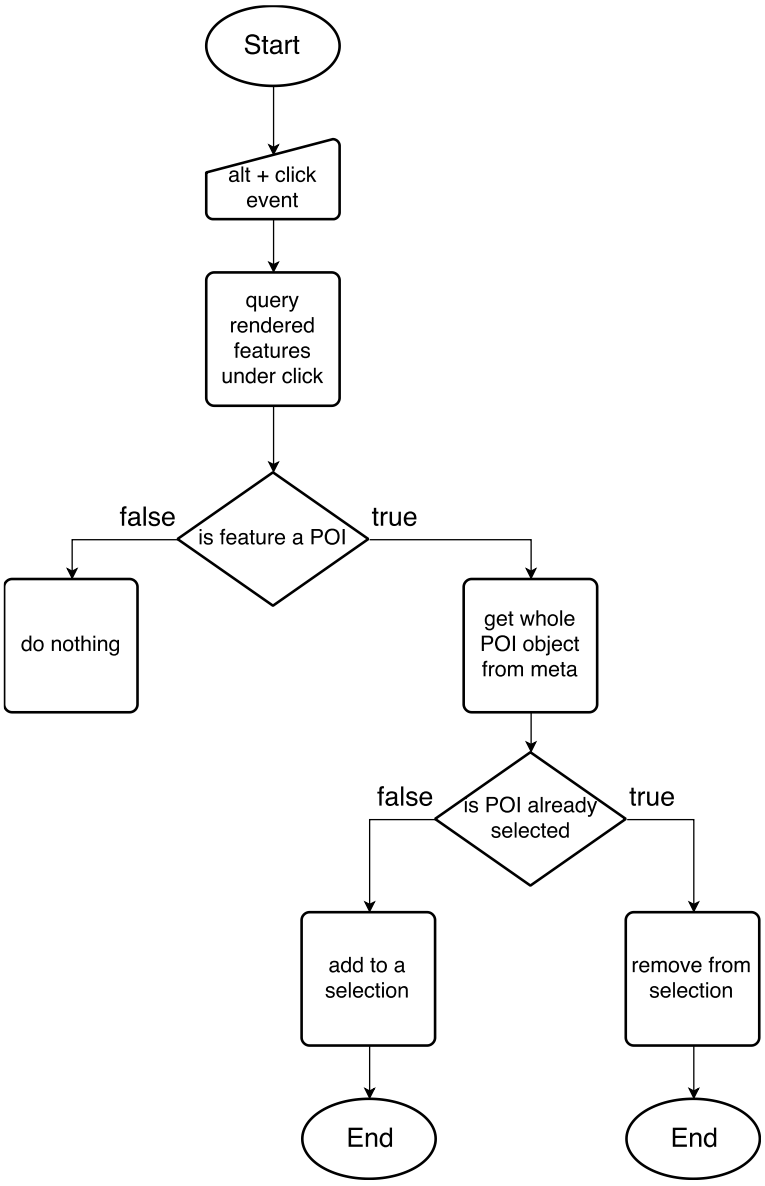


Figure 3. Flow chart presenting alt + click selection

The reason why toggling the selection was combined with the Alt key instead of control was because Mapbox utilizes control key in map rotation interactions. Toggling the selection of objects can also be done from the toolbar's menu of edited POIs.

Another combination of mouse and keyboard event that can trigger multiple selection is combining the shift key to mouse drag. Commonly this is known as box select. By default, Mapbox uses this interaction for a box zoom Feature but their API allows it to be disabled. When shift is hold while dragging the map the editor will register to listen the mouse down and mouse up events and query the drawn bounding box for POI data. An example of box select is presented in Image 8.

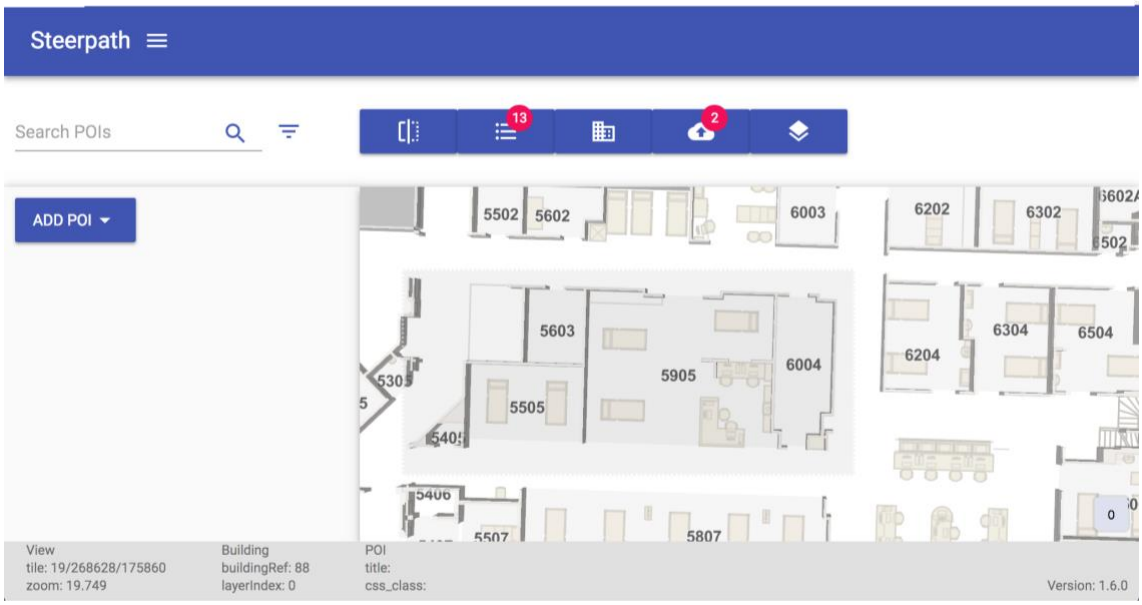


Image 8. Selecting all the rooms from the middle with box selection.

After the user cancels the event either by lifting the mouse click or shift key the editor will select all the features that were inside the bounding box (Image 9.).

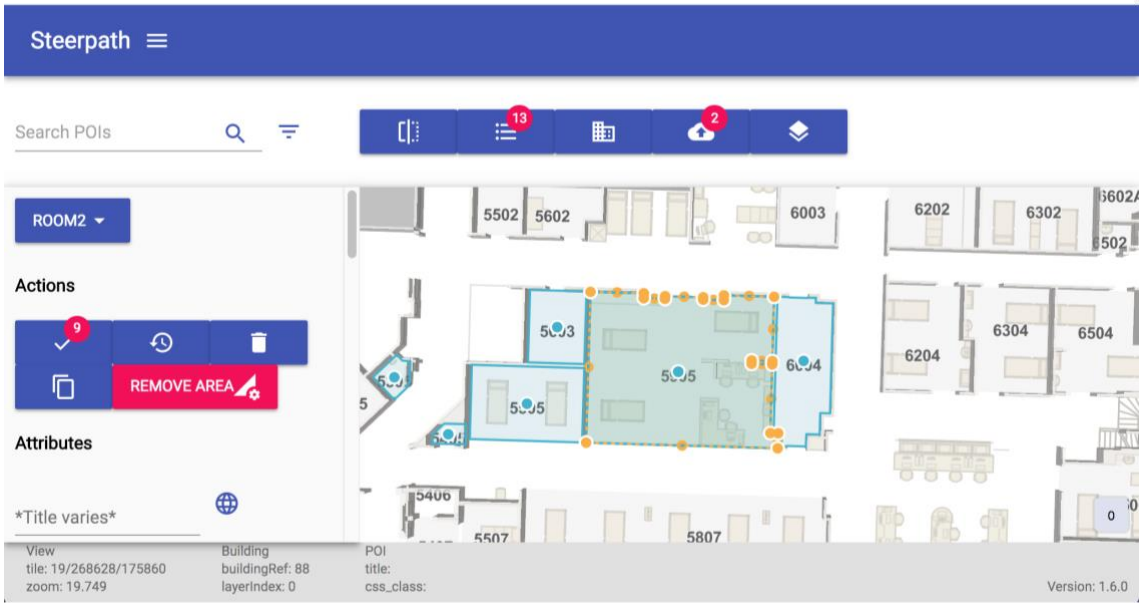


Image 9. All POIs that were inside the bounding are brought to the selection.

This box selection is only available when there are no features in the selection because when the application switches to edit mode the Draw disables the box selection feature. When the Draw is active, the bounding box will work as a selector for which features

should be selected for the drawing. This allows multiple geometries to be edited simultaneously.

When there's a POI in the selection users can select similar type of POIs as the ones in selection. Selecting similar POIs can be done from the sidebar and the selection can be adjusted to meet the user's needs. Users can filter whether they want to select similar POIs from the current viewport, from the list of already edited features, from the current active floor or building or either globally. Selecting similar objects for the selection will query the list of already edited features and server for customer data. Each POI in the list is compared to the ones in selection and if their class match the POI is brought to the edit. Selecting similar POIs can be done as long as the user has only the same class of POIs in the current selection.

Querying the database from POIs can also be accomplished from the toolbar's search input. The input value typed in the search bar can be compared to all of the customer's POI's properties or the it can be filtered to match only certain properties. Search can be an efficient way of finding POIs from the map and edit them. For example, if a customer knows that a certain POI exists in the map it can searched and brought to the edit from the search' result menu.

6.4 Shortcut Commands

Shortcut commands are used to provide quick and easy way to execute actions in applications. Shortcut commands are accessed by combining a modifier key with another key from the keyboard. Shortcut commands were one of the core requirements of the application (Req10). Modifier keys can be Alt, Ctrl, Shift or in Apple computers Command key. By holding a modifier key and pressing a specific key in a keyboard user can perform actions more efficiently. (Computer Hope, 2017.) The basic shortcut keys that were utilised in the map editor are shown in Table 4.

Table 4. List of keyboard shortcut combinations and actions they perform.

Keyboard command	**Action**
enter	Confirm changes that are done and change mode.

escape	Cancel changes that are done and change mode.
backspace	Reset changes and restore POI state
cmd/ctrl + x	Delete POIs in selection
cmd/ctrl + s	Confirm changes and upload them to the server
cmd/ctrl + c	Copy features in selection
cmd/ctrl + v	Paste features in selection
alt + click	Toggle selection and deselection of POI.
shift + click + drag	Box selection.

For attaching actions to keyboard commands, a npm package called combokeys was used. While there are differences between the modifier keys depending on the platform, combokeys allows binding the same actions to different modifiers with ease.

6.5 Real-Time Visualisation of Map Data Changes

As stated in the user requirement survey analysis one of the drawbacks of the current maintenance process is the time that it takes to see changes appear on the final maps after they have been done to the source data (Req8). In order to create a faster feedback loop for the customer the Indoor Map Editor should show the changes that are being done in real time. The process of UpdateManager features is presented in Figure 4.

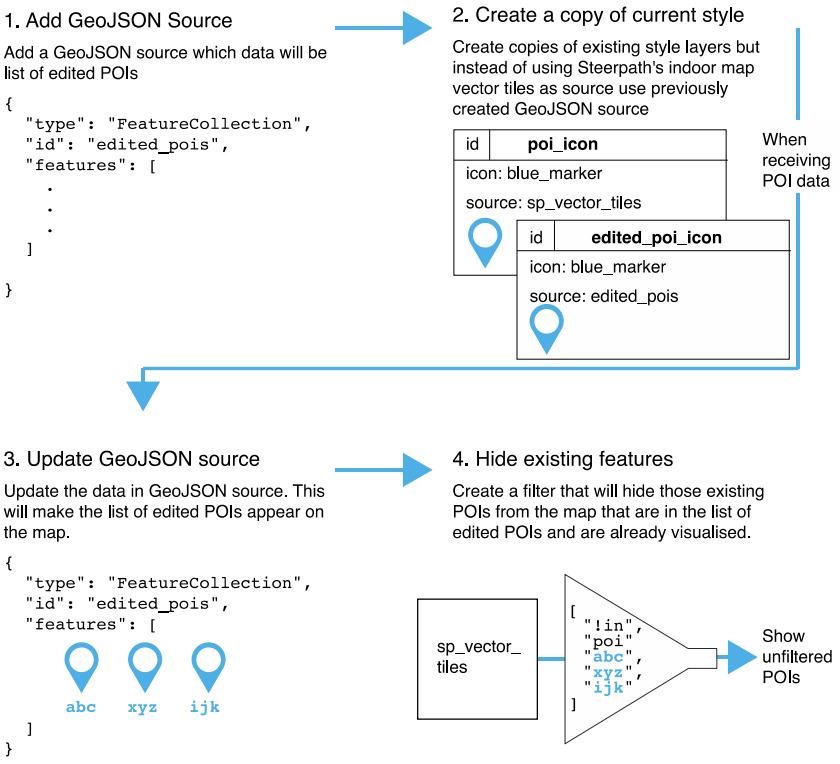


Figure 4. Illustration of how UpdateManager handles showing changes real time.

When an existing Feature is selected it is added to list of edited POIs. This list of edited POIs works as the GeoJSON data that is added to the map (1.). As described previously in order to show sources on Mapbox map they need to have style layers referring to them. In order to make the list of edited POIs appear on the map the same way as the existing ones do, a copy of the map's style is created (2.). Each of the existing style layers that use Steerpath indoor map data as source is copied to the map. The copied layers will refer to the GeoJSON source of list of edited POIs. This way the edited features appear to the user similarly as they did before they were selected.

When an edited POI is presented in the map from the list of edited features it is no longer intuitive to show the user the original unchanged copy of the POI object (3.). Therefore, all of the existing POIs that are selected and edited need to be filtered out from the maps original vector data (4.). The filtering is done based on the POI's identifiers which are added into the original maps style layers that are referring to them. New added POIs are also appended to the list of edited features but since they don't exist in the vector data

and their ids are mapped differently they don't require extra work in regarding showing changes real-time.

6.6 Changing the Visual Appearance of POI

One of the requirements for the Indoor Map Editor was to be able customise POIs visual appearance (Req6). As mentioned previously Mapbox data and style are bound together in a way that style rules dictate how map data should presented. In order to make POI appear differently the editor writes properties to the POI object that will override the default style ruling instead of modifying the style object itself. Binding style object's values directly to features properties is called data-driven styling in Mapbox.

The first visual element that was made fully customizable, was a POI's area colour which is presented in Image 10. In the editor the colour selector was done using a npm package called react-color, which provides ways of selecting colours from a palette or colour picker. The values received from the component are stored as specified attributes in the POIs properties and the values of these attributes are then matched in the data driven style rules.

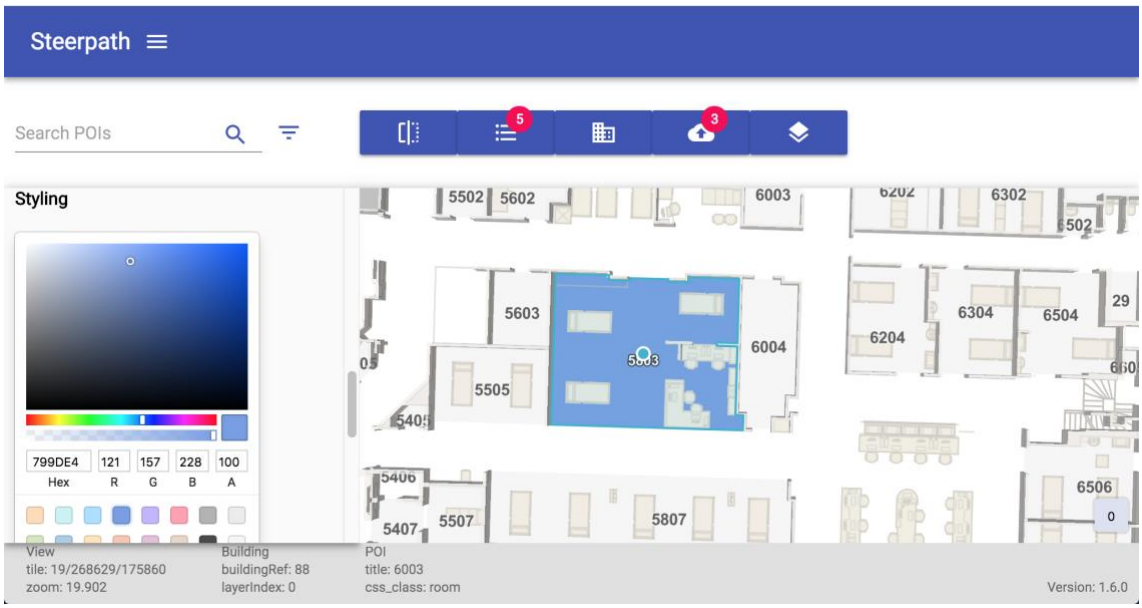


Image 10. Styling POI's area color.

Modifying only the colour of POI's area might create mismatch with rest of the POIs visual elements such as the icon and text colour. Therefore, it was important that in the future

Indoor Map Editor would support customizing rest of the POIs layout and paint style properties. This is also mentioned as one of the future enhancements.

6.7 Showing Detailed Map Data

As stated in the analysis of user requirements survey one of the drawback of the current map maintenance process is the uncertainty how changes that are done to the map will be presented in the final map presentation and how error prone the system is. For example, misspelling a POI's class name in the source will make it not appear when using the default styling. In the survey some of the employees mentioned how Maputnik and Mapbox studio provide ways of showing all of the map data which not visualised normally in a translucent way. Showing objects in translucent way will also help in situations where there are map objects are lying on top of each other (Req12).

In the Indoor Map Editor this feature was referred as X-ray mode (Image 12.). In practise the X-ray mode includes three additional style layers that are filtered to show all the polygon, line and point data in the map in translucent way. X-ray mode will also use Mapbox's feature to show tile coordinates and tile bounds in the map. With mouse right click users can open popup on the map that will list all the features that are in the click coordinates. User can also select POIs from the popup's list. X-ray mode was one part of the concept of how the editor would reveal more relevant map data to the user.

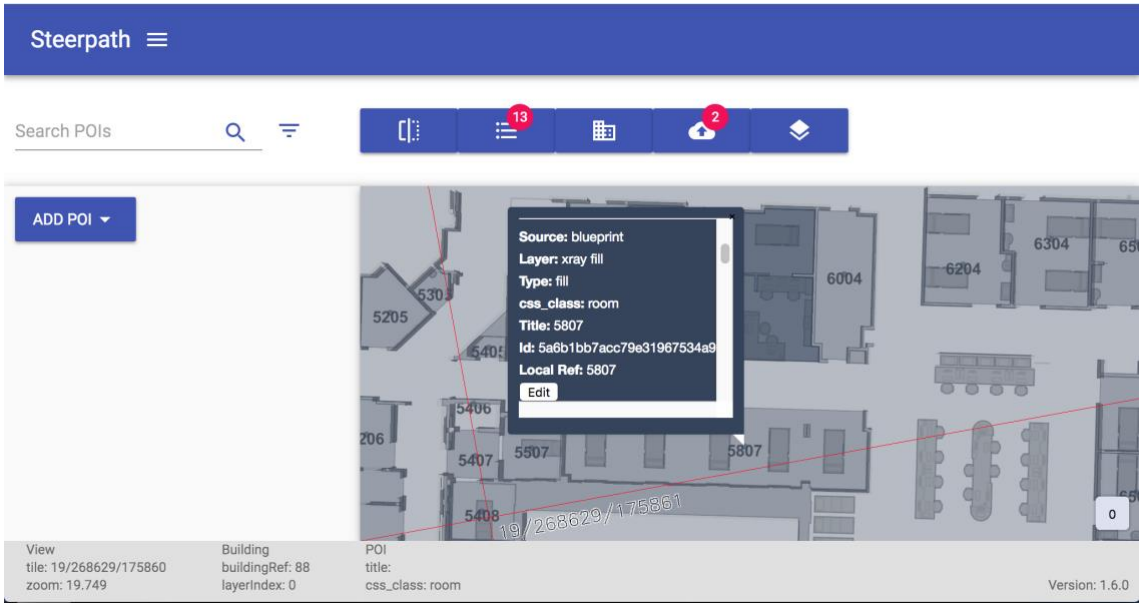


Image 11. Using X-ray mode to show map's tile boundaries and showing POI information from a popup.

In Draftsight one part of the UI is a console bottom bar which gives user info about canvas view and mouse position. In map editor the bottom bar shows the users the tile coordinates under the mouse, zoom level, selected building and if there is a POI under the mouse pointer that can be selected for the edit.

6.8 Future Enhancements

Some of the features mentioned in the user requirements survey which were not implemented into the initial version of Indoor Map Editor were minor low priority features or they were partially implemented as a proof of concept. For example, in the future the Indoor Map Editor should allow further visual customization to the POI. Currently the concept of writing data-driven style attributes to the POI was tested with the POI's area color attribute but in the future the editor and map's style would support more data-driven styling attributes.

Other requirements that was partially implemented was editing the area polygon of POI. Currently the Indoor Map Editor allows modifying the area by moving the polygon or dragging its vertices. In the future the editor would have larger variety of ways to edit POI's geometries. The ability to rotate and scale the existing geometries are features that relatively straightforward to implement since they mainly require calculating new

coordinates from the existing ones. Drawing the POI area freely would make editing easier and more efficient to use, but in order to make the user experience intuitive it will require some additional plugins. Snapping the drawn line along the existing geometries and supporting for orthogonal suggestion while drawing is some of the plugins that need to be implemented to the free draw mode in the future.

One larger feature that will be added in the future will be supporting undoing and redoing commands. There are existing solutions and practices of how this can be done in web applications but in order to utilize those solutions the architecture of the application need to be modified a little. The core concept of undo and redo is to have two lists: past and future to which each of the state of the edited POIs are stored and one object where the present state is stored. Commands undo and redo will then move the states in these lists and change the present state to which the Indoor Map Editor component will refer to.

As mentioned in the summary of the user requirements survey features regarding changing beacon, route or visual data are going to be developed as standalone applications since each of them serve a different role in the company's indoor location and map ecosystem. While some of the practices of the other editors are similar their development was not addressed in this thesis in order to keep the scope manageable. Some of the solutions that were proven to work will be implemented in the other editors as well.

7 Evaluating Application's Usability

To measure how well the application meets the given requirements a usability test was carried out. Three employees participated on the test and the test was conducted at the Steerpath office. The questionnaire can be found in Appendix 4 and the results in Appendix 5.

7.1 The Usability Test

The test followed the common principles and practises of Usability Testing and it consisted of three parts. During the test notes were taken of the users' actions and their use of the application was captured with a screen recording. The usability of the product

can be analysed based on the feedback of the users and how well the application managed to perform the tasks. (Drupal, 2016.)

First part of the test was Pre-Session Questions. In the first part users were introduced to the test and explained why it was being conducted (Drupal, 2016.). The questions in the first part aimed to find out what expectations users have towards the application and how do they expect it to work.

In the second part of the test employees were given tasks to perform with the application. These were based on real map changes which customers most commonly want to be done the maps. There were 7 tasks that employees had to perform and after each task they were asked to describe how they managed to complete it and how their user experience was. The tasks started simple but as the test progressed so did the tasks became more challenging. The task that were in the test were the following:

- 1. Find and edit single POI. In the first task users had to find a single POI and change its title and class from a restaurant to a cafe.
- 2. Adding an area to a POI. Second task was about managing the area of POI. In the task users were asked to add an area to an existing room POI and have it follow the room's walls.
- 3. Add a new POI to the map. As stated in the user requirement survey one of the most common map changes was adding a POI indicating that an area inside the building is under construction. This use case was also tested in the third task where users had to cover an existing POI with a construction area POI.
- 4. Edit Multiple POIs. In the fourth task users were asked to edit multiple POIs simultaneously. In the tasks users were asked to select all the toilet POIs from the map, remove the title from each of them and add a new tag.
- 5. Delete multiple POIs. In order to test multiple editing even further the fifth task asked users to delete multiple POIs from the map. The POIs that needed to be deleted were all the same class.
- 6. Find a Missing POI. In the sixth task users were asked to find a missing POI from the map. For this task only, the floor and one property were given as information about the POI.
- 7. Make Changes Go Live. In order to make changes go live they needed to be uploaded to the server and tiles needed to be regenerated from the data. In the seventh task users were asked to perform these two steps.

The last part of the test was Post-Session Question. In the Post-Session question part users had opportunity to evaluate the product and give feedback about their overall experience. Post-Session Question part consisted of five questions in which users could:

- describe their overall experience with the product
- measuring the usability on a scale from 1 to 5
- list what they liked the most
- list what they liked the least
- and compare the product to the current system.

After the test was conducted the results were analysed.

7.2 Reviewing the Results

Pre-Session Questions

Regarding the expectations towards the product there was a consent among the employees. Everyone expected the application to be fast and easy to use and hoped that it would fasten the map update process. Reliability was also mentioned as one of the requirements.

Tasks

1. Find and edit single POI

While finding the POI, selecting it and changing the title was relatively straight forward some users struggled with changing the class. Only one employee needed assistance on the task with changing the class of the POI. Based on the feedback the dropdown from which POI's class can be changed should have some sort of indicator for what it does instead of stating the class name. Also, even though the POIs are sorted alphabetically in the list the menu could be longer in order to see more possible POI classes. Employees appreciated how Indoor Map Editor automatically fills sidebar's tags based on the POI's class.

2. Add an area to a POI

The area was added to the POI from the sidebar and by dragging the vertices users managed to edit the area with ease. Some users had problem selecting the POI from the map by clicking the room's title. For some reason Mapbox's method

queryRenderedFeatures couldn’t return data in that symbol layer but the users found other ways to select POI instead. One user used X-ray mode to select the POI and one user used search and selected the POI from the results. The way confirmation of editing was done varied between the users. Some users had adapted to pressing the enter key for confirmation and some users switched between clicking around the map or the sidebar for confirmation. Each of the participants managed to complete the task without any assistance.

3. Adding a new POI to the map

Adding a new from the dropdown and making the area to follow the existing POI was clear for the users. One user found an efficient approach by duplicating the POI that was supposed to be covered with copy command. After copying the POI, the user pasted the POI on to the same spot and changed the duplicates class to match construction area POI. Other employees added POI from the sidebar and modified its area manually.

4. Edit multiple POIs

In the task users had different approaches for selecting the POIs for editing. One user used search to get all the POI’s from the map while one user used alt key and mouse click for multiple selection. Since the actual edit process doesn’t differ from single to multiple edit the changes were made at ease. This was also appreciated by the employees how the editing procedure doesn’t differ based on the number of selected objects. At the time of the fourth task users already used application’s interactions more fluently. One employee mentioned that the tag selector could have a legend explaining what the colour coding of the selected tags means.

5. Delete multiple POIs

Selecting the POIs was done by using application’s select similar feature and alt-key and mouse click combination. One user stated that having either control or command key combined to the click would be more intuitive. According to the employees the confirmation dialog before the delete was appreciated while the phrasing of the dialog message could’ve been better.

6. Finding a missing POI

All of the users used search for finding the POI while some mentioned that X-ray mode could've also been used. After selecting the POI from the search result menu users knew how to change the class to accordingly. Some users mentioned an improvement which they would've wanted to have with the search: since the POI that is selected from the search menu may not always be visible in the map the application could centre the view to the selected POI automatically.

7. Uploading Changes to Server

In order to make changes go live they needed to be uploaded to the server and tiles needed to be regenerated from the data. In the seventh task users were asked to perform these two steps. Some users mentioned that the application could highlight somehow that after uploading changes to the server tiles need to be regenerated. User's also would've wanted some way of confirming when the tiles have been generated.

7.3 Post-Session Questions

Based on the answers the overall experience of the application was good and intuitive. Users saw that the application would prove beneficial tool in the future for the map changes. When estimating the experience on a scale from 1 to 5 the average of the application was 4.

Features that users liked most were seeing changes instantly on a map, the clean and intuitive UI, adaptivity and how tasks could be done in many ways. User suggested improvements such as how the application could have a tutorial for first time users where all the application's features would be presented. One employee stated that some features were difficult to discover, and a tutorial would make the application easier to adapt to.

When considering Steerpath's current map maintenance process the employees saw the application as a huge improvement. According to one employee the new application saves a lot time and effort when making map changes. When normally it would have required to make changes to the correct CAD file and upload it to the server the new application allowed changes to be done more efficiently. Another employee stated that

tasks done in the usability test might not have worked at the first time since during the current process there's no visual confirmation or feedback of the changes.

7.4 Summary of the Usability Test

The results of the usability test and users' feedback prove that the indoor map data can be managed with the web application. All users managed to perform the tasks and upload their changes to the server. Users found that the application was easy to adapt to and saw that it would ease their map maintenance process.

Users also found room for improvement for the application. Some users mentioned that phrasing at some parts of the application could have been better and more descriptive. In addition to smaller UI related improvements all users mentioned that the application should have a tutorial for the first-time users.

8 Conclusion

The aim of this thesis was to enhance Steerpath's indoor map maintenance process and make it efficient and user friendly. For improving the map update process a web application was planned, implemented and verified. In this thesis there were two main points which were researched. One of the points was to study different web map technologies and how they could be utilised in the application. Another point was to study the user experience domain and design the best possible User Experience for the application. The work was carried out in the following steps:

- Researching the theory how map data can be constructed and presented in web environment.
- Studying company specific backgrounds and data models.
- Reviewing User Experience of the existing and utilised map editors.
- Conducting comprehensive survey among the employees of Steerpath to figure out the requirements for the application.
- Designing the application's User Experience and architecture.
- Implementing requirements and developing the application.
- Conducting a usability test to measure how well the application meets the requirements.

The application was created based on the most common use cases which were received from the user requirements survey. Based on the results and use cases the first implementation of the application was developed. At the end of the project a usability test was held among the employees of Steerpath.

Overall, the application shows that indoor map data can be managed and maintained efficiently and user-friendly way on the web. Based on the results and feedback of the usability test, the application was proven to meet the requirements it was set by the company. The company sees potential in the application and once it was proven to be an efficient tool the design for further development started. While the initial version of the application was narrowed down to handle the POI data on the map, the concepts and practises of the edit procedure and its User Experience can be applied further into other map data editors as well. In future the application will be integrated as part of the Steerpath's indoor map ecosystem.

References

Anttila, Tuikku (2018) [Interview] (26 February 2018)

Bernstein, Gregg (2015) *How To Create UX Personas* Available at: https://uxmastery.com/create-ux-personas/ (Accessed 3 February 2018)

Bostock, Mike (2016) *TopoJSON* Available at: https://github.com/topojson/topojson/wiki (Accessed 27 February 2018)

Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S., Schaub, T. (2016) *The GeoJSON Format* Available at: https://tools.ietf.org/html/rfc7946 (Accessed 26 February 2018)

Computer Hope (2017) *Computer keyboard shortcut keys* Available at: https://www.computerhope.com/shortcut.htm (Accessed 18 February 2018)

Dempsey, Caitlin (2018) *What is GIS?* Available at: https://www.gislounge.com/what-is-gis/ (Accessed 25 February 2018)

Drupal (2016) *Tutorial how-to conduct usability testing* Available at: https://www.drupal.org/docs/develop/usability-testing/tutorial-how-to-conduct-usability-testing (Accessed 17 March 2018)

Gackenheimer, Corym (2015) [Online] Available at: https://books.google.fi/books?hl=fi&lr=&id=NZCKCgAAQBAJ&oi=fnd&pg=PR6&dq=facebook+react+framework&ots=KAsyRmHy0h&sig=f_vdZlHHReiOQggC0e152-u8DgI&redir_esc=y#v=onepage&q&f=false (Accessed 18 February 2018)

Goswami, Subrata (2013*) Indoor Location Technologies. Springer* [Online] Available at: https://link-springer-com.ezproxy.metropolia.fi/book/10.1007%2F978-1-4614-1377-6 (Accessed 26 November 2017)

Haveri, Merja, Huhtala, Jussi, Häkkilä, Jonna, Puikkonen, Arto, Sarjanoja, Ari-Heikki (2009) 'Towards designing better maps for indoor navigation: experience from a case study' *Proceedings of the 8th International Conference on Mobile and Ubiquitous Multimedia Article No. 16* [Online] Available at: https://dl.acm.org/citation.cfm?id=1658566 (Accessed 10 April 2018)

Langley, Peter J. (2016) *OpenLayers 3.x Cookbook* [Online] Available at: https://books.google.fi/books?hl=fi&lr=&id=5JjjCwAAQBAJ&oi=fnd&pg=PP1&dq=openlayers+&ots=VMo0bGVPgD&sig=Qm5lr2t_BYtXY6B342KDbQGw-C8&redir_esc=y#v=onepage&q=openlayers&f=false (Accessed 18 February 2018)

LearnOSM (2016) *The iD Editor* [Online] Available at: http://learnosm.org/en/beginner/id-editor/#id-editor-user-interface (Accessed 9 March 2018)

MacWright, Tom (2015) *More than you ever wanted to know about GeoJSON* Available at: https://macwright.org/2015/03/23/geojson-second-bite (Accessed 14 January 2018)

Mapbox (2018) *About Mapbox* Available at: https://www.mapbox.com/about/ (Accessed 13 January 2018)

Mapbox (2018) *Maps* Available at: https://www.mapbox.com/maps/ (Accessed 14 January 2018)

Mapbox (2018) *Mapbox Vector Tile Specification* Available at: https://www.mapbox.com/vector-tiles/specification/ (Accessed 22 January 2018)

Mapbox (2018) *Mapbox Style Specification* Available at: https://www.mapbox.com/mapbox-gl-js/style-spec (Accessed 22 January 2018)

Moreno, Helga *The Gap between UI and UX Design – Know the Difference* Available at: https://onextrapixel.com/the-gap-between-ui-and-ux-design-know-the-difference/ (Accessed 4 February 2018)

Ordnance Survey (2018) *Raster and Vector data* Available at: https://www.ordnancesurvey.co.uk/support/understanding-gis/raster-vector.html (Accessed 14 January 2018)

OpenStreetMap (2018) *OpenStreetMap* Available at: https://www.openstreetmap.org/ (Accessed 23 April 2018)

Pichel, Roman (2016) *10 Tips for writing good user stories* Availabale at: https://www.romanpichler.com/blog/10-tips-writing-good-user-stories/ (Accessed 3 February 2018)

Soegaard, Mads (2018) *Usability: A part of the User Experience* Available at: https://www.interaction-design.org/literature/article/usability-a-part-of-the-user-experience (Accessed 3 February 2018)

Scerri, Darren (2017) *Demystifying Babel and ES6* Available at: https://medium.com/@darrenscerri/demystifying-babel-and-es6-98a9e6624573 (Accessed 19 February 2018)

Taraldsvik, Mats (2011) *Exploring the Future: is HTML5 the solution for GIS Applications on the World Wide Web?* [Online] Available at: http://mats.taraldsvik.net/gd12/publications/html5andgis.pdf (Accessed 18 February 2018)

Temprano, Victor Gerard (2016) *Google Maps API or Leaflet: What's Best for your project?* Available at: https://www.codementor.io/victorgerardtemprano/google-maps-api-or-leaflet--what-s-best-for-your-project-faaev60vm (Accessed 21 February 2018)

Virkkilä, Mikko (2018) [Interview] (February 21 2018)

Wanyoike, Michael (2017) *A Beginner's Guide to npm – the Node Packaging Manager.* Available at: https://www.sitepoint.com/beginners-guide-node-package-manager/ (Accessed 12 February 2018)

Webpack (2018) *Concepts* Available at: https://webpack.js.org/concepts/ (Accessed 18 February 2018)

Xia, Vincent (2017) *UX Is Not UI: What Is The Difference between UX and UI Design* Available at: https://medium.com/@Vincentxia77/ux-is-not-ui-what-is-the-difference-between-ux-and-ui-design-by-devin-f3e4932e738f (Accessed 3 February 2018)

User Requirements Survey - Questionnaire

Map editor use cases and requirements

This questioner will help us track what are the most common use cases and requirements for the new map editor. Your answers will also be used when I write my thesis but they will be treated anonymously. If you don't want your answers to be used in my thesis please let me know.

1. **What are the most common changes customers want to be done to the maps?**

2. **What are the required procedures in order to make these changes?**

3. **Are there some good features or practises with any of the existing tools/editors (f.e DraftSight, OpenStreetMap Editor, Maputnik editor or Mapbox Studio) that you would like to have in the new editor?**

4. **Please paste here copy of your most recent DraftSight commands. You can do this by right clicking Command Window and selecting "copy history" If you haven't used Draftsight in a while you can send me the commands later.**

User Requirements Survey - Results

Timestamp	What are the most common changes customers want to be done to the maps?	What are the required procedures in order to make these changes?	Are there some good features or practises with any of the existing tools/editors (f.e DraftSight, OpenStreetMap Editor, Maputnik editor or Mapbox Studio) that you would like to have in the new editor?	Please paste here copy of your most recent DraftSight commands. You can do this by right clicking Command Window and selecting "copy history" If you haven't used Draftsight in a while you can send me the commands later.
1/5/2018 1	Most common: add or remove POIs, edit attributes of existing POIs. Sometimes there are requests for style modification but actually pretty rarely. Mostly things like a POI should appear at a lower/higher zoom level, or an icon should be added or changed. There are routing change requests as well.	If I have the original CADs, I'll edit them directly. Depending on the customer I'll then upload to Meta from my own computer, generate tiles and routes, or if possible upload to steerpath-blueprint.herokuapp.com which will do the same steps automatically. Sometimes I'll need to generate offline bundles as well. If it's a simple change to an existing POI or adding new POIs I'll make the changes via Meta API. For Changi I can use Confidex Console.	In I like the ability to search POIs with tags and edit multiple at the same time (not sure if it worked though), for example add a new tag to many POIs at once. Draftsight: select by drawing a box. Maputnik: seeing the style in json format and being able to edit that directly, though it's a bit out of scope here.	
1/5/2018 1	Most commonly: add or remove points of interest (POIs) or edit their attributes. Sometimes larger changes such as removing part of a floor since it's not publicly accessible, but mostly smaller edits. Style changes are requested less often, but sometimes there's a need for new icons for example. For some customers routing changes are common.	Most times I edit the original CAD, then process it on my computer, upload the data to our database, generate map tiles and routes. We have set up a server that does the processing automatically and I can use it for some customers' data but I still need to make the changes in the CADs. Since the processing is slow and it has to complete before I can check the result, even small changes can take a lot of time. For small changes like editing the title of a POI I edit the data via our REST API.	In I like that you can search POIs with tags and edit multiple at the same time. In CADs with lots of POIs it's hard to narrow down the search for the correct POI otherwise. Multiple edits at the same time are useful because lots of changes are of the type 'add a new tag to all POIs in this area'. In Draftsight, I like that you can select by drawing rectangles, and add to select by clicking. In Maputnik I like the style editor that shows the JSON form of the style and allows you to edit it, but style edits are a bit out of scope here. Maybe a nice feature could be a geojson view to the POI - that could make it easier to edit area geometries for example. But on the other hand it would then be easier to post invalid data.	
1/5/2018 1	Point of interest names, Adding point of interests (icons & areas), adding under maintenance areas that disable routing through those area + show under maintenance area on the map		Copy & paste, undo. Having visual confirmation (preview) to changes immediately is important when editing parameters or changing element type. Selecting multiple elements to edit their parameters at once is very commonly used feature. Qselect objects -> edit parameters -> unselect	

Timestamp	What are the most common changes customers want to be done to the maps?	What are the required procedures in order to make these changes?	Are there some good features or practises with any of the existing tools/editors (f.e DraftSight, OpenStreetMap Editor, Maputnik editor or Mapbox Studio) that you would like to have in the new editor?	Please paste here copy of your most recent DraftSight commands. You can do this by right clicking Command Window and selecting "copy history" If you haven't used Draftsight in a while you can send me the commands later.
1/5/2018 1	-	-	1. qselect -> match layer 2. would like to have functionality when selecting one object etc. and then "select all from same layer" !!! 3. like 2 but also option "show only selected object's layer"	
1/8/2018 1	Poi names changes and beacon location changes.	Edit the cad with the new details or beacon location. Save and upload to regenerate NDD and publish map changes.	Selecting many pois or beacons and making a group change to name or ID etc.	
1/8/2018 1	Changing/adding retail and restaurant locations, moving beacons	Receive the request and logging it in a appropriate excel file, finding the right CAD, finding the right location in the cad, making changes and hoping for Draftsight not to crash again. Saving changes, seeing the results and possibly doing more changes. Committing to git. Informing the customer that the request is processed and update the excel regarding the task.	Ability to create and adjust organic shapes. New feature that I want: Functional version control, backup files and fast load times. Catastrophic crash feature not needed.	

User Requirements Survey - DraftSight Commands

Employee 1	Employee 2	Employee 3	Employee 4	All	Unique Commands	Count		Sorted	Count
: Opening "R2013" drawing				: Opening "R2(	: Opening "R2013" draw	4		<Switching to: Model>	7799
:		:		:	:	2741		:	2741
:		:		:	: _LAYERPREVIEW	2		Specify next vertex»	882
: _LAYERPREVIEW		:		: _LAYERPREVI	: _MAINSELECTSpecify o	733		: _MAINSELECTSpecify opposite	733
:		: pedit		:	: _MAINSELECT	636		Options: Arc, Close, Halfwidth, Ler	729
: _MAINSELECTSpecify o		EDITPOLYLINE		: _MAINSELEC	Options: active Layer or	92		: _MAINSELECT	636
: _MAINSELECTSpecify o		Options: Multiple		: _MAINSELEC	Specify destination»	94		: «Cancel»	313
:		Specify polyline»		:	Automatic save to /Users/tu	9		Options: Enter to continue from las	233
:		Specify entities»		:	: _OPEN	1		Options: Arc, Halfwidth, Length, Ur	211
: _LAYERPREVIEW		330 found		: _LAYERPREVI	: move	4		Specify start point»	209
: _MAINSELECTSpecify o		Specify entities»		: _MAINSELEC	1 found	147		1 found	147
: _MAINSELECTSpecify o		Default: Yes		: _MAINSELEC	Default: Displacement	4		All layers have been turned on.	147
: _MAINSELECTSpecify o		Confirm: Convert		: _MAINSELEC	Options: Enter to use from	4		Undo SHOWLAYERS	144
:		Specify Yes or N		:	: _PASTE	5		Undo INTELLIZOOM	128
: _MAINSELECTSpecify o		Options: Close, D		: _MAINSELEC	STRETCH	46		: POLYLINE	110
: _MAINSELECTSpecify o		Specify option» «		: _MAINSELEC	Options: Base point, Copy,	46		Specify destination»	94
: _MAINSELECT		: _MAINSELECT		: _MAINSELEC	Stretch point»	45		Options: active Layer or	92
: _MAINSELECT		: «Cancel»		: _MAINSELEC	: '_LAYER	1		STRETCH	46
: _MAINSELECTSpecify o		: _MAINSELECT		: _MAINSELEC	Options: Displacement or	4		Options: Base point, Copy, Undo, «	46
: _MAINSELECT		: «Cancel»		: _MAINSELEC	Specify from point»	4		Stretch point»	45
:		:		:	: _SAVEAS	1		Specify next vertex» «Cancel»	31
Options: active Layer or		:		Options: active L	: _DELETE	10		Undo DELETE	26
Specify destination»		:		Specify destinati	1 found, 3 total	5		Undo Modify Property	24
:		:		:	4 found	2		: _MAINSELECTSpecify opposite	22
:		:		:	1 found, 2 total	7		Specify next vertex» _Undo	22
:		:		:	2 found	4		Specify start point» «Cancel»	20
Automatic save to /Users/tu		Options: Enter to		Automatic save t	Press ESC or ENTER to ex	1		: POLYLINE	16
:		Specify start poir		:	Resuming MAINSELECT c	1		Active settings: Entity Style=No La	16

:		Options: Arc, Clo		:	Specify opposite corner:_U	1		Nothing to redo	15
:		Options: Arc, Clo		:	Invalid point.Specify opposi	1		: U	13
:		Specify next vert		:	: _U	1		Automatic save to C:\Users\Steerp	12
:		Specify next vert		:	Undo INTELLIZOOM	128		Redo INTELLIZOOM	11
Automatic save to /Users/tu		: Opening "R201		Automatic save t	Undo PAN_INTERNAL	2		: _DELETE	10
:		:		:	: _SAVE	1		Undo LAYERPREVIEW	10
:		:		:	: pedit	1		Automatic save to C:\Users\Steerp	10
:		:		:	EDITPOLYLINE	1		Automatic save to /Users/tuukkaail	9
:		Options: active L		:	Options: Multiple or	1		Automatic save to C:\Users\Steerp	8
:		Specify destinati		:	Specify polyline» m	1		1 found, 2 total	7
:		:		:	Specify entities»	5		Specify layer by selecting entity»	7
:		:		:	330 found	1		Undo SNAPGRIP	6
:		Options: active L		:	Default: Yes	1		All commands redone.	6
:		:		:	Confirm: Convert Lines and	1		: qselect	6
:		:		:	Specify Yes or No» y	1		SMARTSELECT	6
:		:		:	Options: Close, Decurve, F	1		: _PASTE	5
:		Options: active L		:	Specify option» «Cancel»	2		1 found, 3 total	5
: _OPEN		:		: _OPEN	: «Cancel»	313		Specify entities»	5
:		:		:	Options: Enter to continue f	233		: Already zoomed in as far as poss	5
: move		Options: active L		: move	Specify start point»	209		: GROUP	5
1 found		Specify destinati		1 found	Options: Arc, Close, Halfwi	729		Undo POLYLINE	5
Default: Displacement		:		Default: Displace	Specify next vertex»	882		Options: Undo, Enter to exit or	5
Options: Enter to use from p		Options: active L		Options: Enter to	Automatic save to C:\Users	1		: Opening "R2013" drawing file	4
Specify destination»		Specify destinati		Specify destinati	Automatic save to C:\Users	8		: move	4
Automatic save to /Users/tu		:		Automatic save t	: _MAINSELECTSpecify o	22		Default: Displacement	4
:		:		:	Undo PASTE	3		Options: Enter to use from point as	4
: _PASTE		:		: _PASTE	Specify reference point»	1		Options: Displacement or	4
Options: active Layer or		Automatic save t		Options: active L	49 found	1		Specify from point»	4

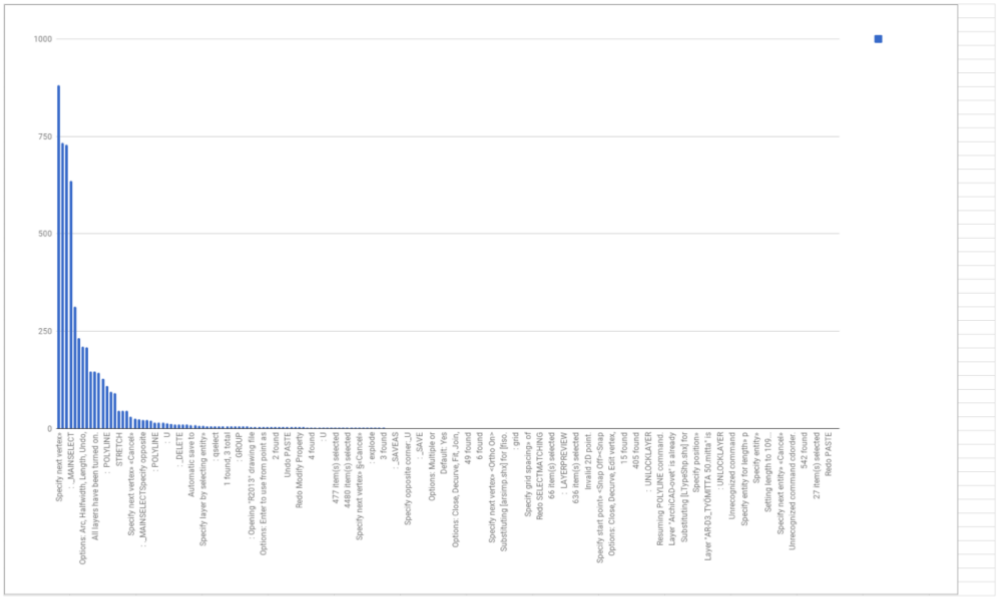
Specify destination»		:		Specify destinatic	Options: Arc, Halfwidth, Ler	211		2 found	4
: _MAINSELECT		:		: _MAINSELECT	: POLYLINE	16		Automatic save to C:\Users\Steerp	4
:		:		:	Undo DELETE	26		Specify next entity»	4
: _MAINSELECTSpecify op		:		: _MAINSELECT	Specify entities» «Cancel»	1		Undo PASTE	3
: _MAINSELECTSpecify op				: _MAINSELECT	Specify destination» «Canc	1		Automatic save to C:\Users\Steerp	3
:		: Opening "R201		:	All layers have been turned	147		146 item(s) selected	3
STRETCH		:		STRETCH	Undo SHOWLAYERS	144		Redo Modify Property	3
Options: Base point, Copy,		:		Options: Base pc	Specify start point» «Cance	20		Undo UNLOCKLAYER	3
Stretch point»		Automatic save t		Stretch point»	Undo SNAPGRIP	6		: _LAYERPREVIEW	2
:		:		:	: U	2		4 found	2
:		:		:	0 found, 1 total	2		Undo PAN_INTERNAL	2
: _MAINSELECTSpecify op		: _MAINSELECT		: _MAINSELECT	6 found	1		Specify option» «Cancel»	2
:		:		:	Stretch point» <ESnaps Off	1		: U	2
:		:		:	Specify start point» <ESnap	1		0 found, 1 total	2
:		:		:	Specify next vertex» <Ortho	1		9 item(s) selected	2
:		1 found		:	Specify next vertex» «Canc	31		477 item(s) selected	2
:		:		:	5 found	1		Undo SELECTMATCHING	2
: _PASTE		:		: _PASTE	: Already zoomed in as far a	5		Redo DELETE	2
Options: active Layer or		:		Options: active L	Automatic save to C:\Users	1		4480 item(s) selected	2
Specify destination»		Options: active L		Specify destinatic	Substituting [arsimp.shx] fo	1		Default: 0	2
: _MAINSELECT		Specify destinatic		: _MAINSELECT	Automatic save to C:\Users	3		Tab Index» 3	2
: _MAINSELECTSpecify op		: _MAINSELECT		: _MAINSELECT	Active settings: Entity Style	16		Specify next vertex» §«Cancel»	2
:		: «Cancel»		:	9 item(s) selected	2		Specify entities to weld to source»	2
: _MAINSELECTSpecify op		: _MAINSELECT		: _MAINSELECT	3 item(s) selected	1		1 was on a locked layer.	2
:		: _MAINSELECT		:	<Switching to: Model>	7799		: explode	2
:		: «Cancel»		:	Rebuilding viewports...	1		Undo EXPLODE	2
:		: _MAINSELECT		:	: grid	1		Layer "--Steerpath wall" has been	2
: _PASTE		:		: _PASTE	Default: 2400	1		3 found	2

Stretch point»		:		Stretch point»	Options: drawing Bounds, r	1		: _OPEN	1
: _MAINSELECTSpecify op		:		: _MAINSELECT	Specify grid spacing» of	1		: '_LAYER	1
:		1 found		:	Automatic save to C:\Users	4		: _SAVEAS	1
:		:		:	146 item(s) selected	3		Press ESC or ENTER to exit.	1
: _MAINSELECT		:		: _MAINSELECT	146 found	1		Resuming MAINSELECT comman	1
:		:		:	477 item(s) selected	2		Specify opposite corner:_U	1
:		Options: active L		:	477 found	1		Invalid point.Specify opposite corn	1
: _MAINSELECTSpecify op		Specify destinatic		: _MAINSELECT	Undo SELECTMATCHING	2		: _U	1
:		: _MAINSELECT		:	Redo INTELLIZOOM	11		: _SAVE	1
: _MAINSELECTSpecify op		: _MAINSELECT		: _MAINSELECT	Redo DELETE	2		: pedit	1
: '_LAYER		: «Cancel»		: '_LAYER	Redo SELECTMATCHING	1		EDITPOLYLINE	1
: _MAINSELECTSpecify op		:		: _MAINSELECT	All commands redone.	6		Options: Multiple or	1
:		:		:	Nothing to redo	15		Specify polyline» m	1
:		:		:	6166 item(s) selected	1		330 found	1
Automatic save to /Users/tu		Options: active L		Automatic save t	1 found, 4 total	1		Default: Yes	1
:		Specify destinatic		:	66 item(s) selected	1		Confirm: Convert Lines and Arcs to	1
:		:		:	0 found, 124 total	1		Specify Yes or No» y	1
:		Undo PASTE		:	: qselect	6		Options: Close, Decurve, Fit, Join,	1
Options: Base point, Copy,		: _MAINSELECT		Options: Base pc	SMARTSELECT	6		Automatic save to C:\Users\Steerp	1
:		:		:	1693 item(s) selected	1		Specify reference point»	1
Options: active Layer or		:		Options: active L	: LAYERPREVIEW	1		49 found	1
Specify destination»		:		Specify destinatic	Undo Modify Property	24		Specify entities» «Cancel»	1
: _MAINSELECT		1 found		: _MAINSELECT	8221 item(s) selected	1		Specify destination» «Cancel»	1
:		:		:	8221 found	1		6 found	1
: _MAINSELECTSpecify op		:		: _MAINSELECT	4480 item(s) selected	2		Stretch point» <ESnaps Off><ESn	1
:		:		:	Automatic save to C:\Users	12		Specify start point» <ESnaps On>	1
: _PASTE		Options: active L		: _PASTE	636 item(s) selected	1		Specify next vertex» <Ortho On>	1
Options: Displacement or		Specify destinatic		Options: Displac	1 item(s) selected	1		5 found	1

Specify from point»		: _MAINSELECT		Specify from poir	Undo LAYERPREVIEW	10		Automatic save to C:\Users\Steerp	1
:		: «Cancel»		:	Redo Modify Property	3		Substituting [arsimp.shx] for [fiso.s	1
: _MAINSELECTSpecify op		:		: _MAINSELECT	Specify start point» _Toolbo	1		3 item(s) selected	1
:		:		:	Invalid 2D point.	1		Rebuilding viewports...	1
:		:		:	: GROUP	5		: grid	1
Automatic save to /Users/tu		:		Automatic save t	Undo POLYLINE	5		Default: 2400	1
:		:		:	Specify next vertex» <ESna	1		Options: drawing Bounds, match S	1
:		:		:	: '_+DRAFTINGOPTIONS	1		Specify grid spacing» of	1
: _MAINSELECTSpecify op		:		: _MAINSELECT	Default: 0	2		146 found	1
:		:		:	Tab Index» 3	2		477 found	1
:		:		:	Specify next vertex» _Undo	22		Redo SELECTMATCHING	1
:		:		:	Specify start point» <Snap	1		6166 item(s) selected	1
: _MAINSELECTSpecify op		:		: _MAINSELECT	Specify next vertex» §«Can	2		1 found, 4 total	1
: _PASTE		: _MAINSELECT		: _PASTE	: <Snap On>	1		66 item(s) selected	1
:		:		:	: <Snap Off>	1		0 found, 124 total	1
:		:		:	Options: Close, Decurve, E	1		1693 item(s) selected	1
:		:		:	: POLYLINE	110		: LAYERPREVIEW	1
Options: active Layer or		Specify referenc		Options: active L	: U	13		8221 item(s) selected	1
Specify destination»		49 found		Specify destinati	: weld	1		8221 found	1
:		:		:	Specify base entity»	1		636 item(s) selected	1
STRETCH		: _MAINSELECT		STRETCH	Specify entities to weld to s	2		1 item(s) selected	1
: _MAINSELECTSpecify op		: «Cancel»		: _MAINSELECT	15 found	1		Specify start point» _ToolboxMenu	1
:		: _MAINSELECT		:	460 segments added to Pol	1		Invalid 2D point.	1
STRETCH		:		STRETCH	6 item(s) selected	1		Specify next vertex» <ESnaps Off>	1
:		:		:	1 was on a locked layer.	2		: '_+DRAFTINGOPTIONS	1
:		:		:	: explode	2		Specify start point» <Snap Off><S	1
:		1 found		:	405 found	1		: <Snap On>	1
: _MAINSELECTSpecify op		:		: _MAINSELECT	Cannot explode 2 entities.	1		: <Snap Off>	1

:		:		:	Undo EXPLODE	2		Options: Close, Decurve, Edit verte	1
:		:		:	Specify next vertex» <Ortho	1		: weld	1
:		Options: active L		:	Specify layer by selecting e	7		Specify base entity»	1
:		Specify destinati		:	Layer "--Steerpath wall" has	2		15 found	1
:		Automatic save t		:	: UNLOCKLAYER	1		460 segments added to PolyLine	1
:		: _MAINSELECT		:	Specify layer by selecting e	1		6 item(s) selected	1
:		: «Cancel»		:	Specify start point» '_+DRA	1		405 found	1
:		: _MAINSELECT		:	Resuming POLYLINE comm	1		Cannot explode 2 entities.	1
:		:		:	: LAYER	1		Specify next vertex» <Ortho On><(	1
:		:		:	Layer "ArchiCAD-ovet" has	1		: UNLOCKLAYER	1
: _SAVEAS		:		: _SAVEAS	Layer "ArchiCAD-ovet" is al	1		Specify layer by selecting entity» «	1
Options: Base point, Copy,		1 found		Options: Base pc	Substituting [arsimp.shx] fo	1		Specify start point» '_+DRAFTING	1
:		:		:	File not found - 090C-A123	1		Resuming POLYLINE command.	1
Automatic save to /Users/tu		:		Automatic save t	Substituting [LTypeShp.shx	1		: LAYER	1
Stretch point»		:		Stretch point»	Automatic save to C:\Users	10		Layer "ArchiCAD-ovet" has been u	1
:		Options: active L		:	: §	1		Layer "ArchiCAD-ovet" is already u	1
:		Specify destinati		:	Options: Angle, angle Bisec	1		Substituting [arsimp.shx] for [Arial	1
: _MAINSELECT		: _MAINSELECT		: _MAINSELECT	Specify position»	1		File not found - 090C-A1230-00_0	1
:		: «Cancel»		:	Options: Enter to exit or	1		Substituting [LTypeShp.shx] for [0	1
:		: _MAINSELECT		:	Specify next position» «Can	1		: §	1
:		:		:	3 found	2		Options: Angle, angle Bisect, Horiz	1
: _MAINSELECT		:		: _MAINSELECT	Layer "AR-D3_TYÖMITTA	1		Specify position»	1
:		:		:	1 found, 0 total	1		Options: Enter to exit or	1
: _MAINSELECTSpecify op		1 found		: _MAINSELECT	Layer "AR-G_MODUULI.m	1		Specify next position» «Cancel»	1
:		:		:	: UNLOCKLAYER	1		Layer "AR-D3_TYÖMITTA 50.mitta	1
: _MAINSELECTSpecify op		:		: _MAINSELECT	Layer "AR-G_MODUULI.m	1		1 found, 0 total	1
:		:		:	Undo UNLOCKLAYER	3		Layer "AR-G_MODUULI.merkintä"	1
: _MAINSELECTSpecify op		Options: active L		: _MAINSELECT	: chagelength	1		: UNLOCKLAYER	1

: _MAINSELECTSpecify op		Specify destinatic		: _MAINSELECT	Unrecognized command ch	1		Layer "AR-G_MODUULI.merkintä"	1
:		: _MAINSELECT		:	: editlength	1		: chagelength	1
: _MAINSELECTSpecify op		: «Cancel»		: _MAINSELECT	Options: Dynamic, Increme	1		Unrecognized command chagelen	1
:		: _MAINSELECT		:	Specify entity for length» p	1		: editlength	1
: _MAINSELECTSpecify op		:		: _MAINSELECT	Default: 100	1		Options: Dynamic, Increment, Perc	1
: _MAINSELECTSpecify op		:		: _MAINSELECT	Specify percentage» 110	1		Specify entity for length» p	1
:		:		:	Specify entity»	1		Default: 100	1
: _MAINSELECT		1 found		: _MAINSELECT	Setting length to 90...	1		Specify percentage» 110	1
: _MAINSELECT		:		: _MAINSELECT	Options: Undo, Enter to exi	5		Specify entity»	1
: _MAINSELECT		:		: _MAINSELECT	Specify next entity»	4		Setting length to 90...	1
:		:		:	Setting length to 99...	1		Setting length to 99...	1
:		Options: active L		:	Setting length to 109...	1		Setting length to 109...	1
: _DELETE		Specify destinatic		: _DELETE	Setting length to 120...	1		Setting length to 120...	1
1 found, 3 total		: _MAINSELECT		1 found, 3 total	Setting length to 132...	1		Setting length to 132...	1
: _MAINSELECT		: «Cancel»		: _MAINSELECT	Specify next entity» «Cance	1		Specify next entity» «Cancel»	1
:		: _MAINSELECT		:	: _DISPLAYORDER_FRON	1		: _DISPLAYORDER_FRONT	1
:		:		:	: cdorder	1		: cdorder	1
: _DELETE		:		: _DELETE	Unrecognized command cd	1		Unrecognized command cdorder. I	1
1 found		:		1 found	: qselect«Cancel»	1		: qselect«Cancel»	1
: _MAINSELECT		1 found		: _MAINSELECT	542 item(s) selected	1		542 item(s) selected	1
:		:		:	542 found	1		542 found	1
:		:		:	: S	1		: S	1
: _DELETE		:		: _DELETE	1266 item(s) selected	1		1266 item(s) selected	1
1 found		Options: active L		1 found	27 item(s) selected	1		27 item(s) selected	1
: _MAINSELECTSpecify op		Specify destinatic		: _MAINSELECT	Undo MOVE	1		Undo MOVE	1
:		: _MAINSELECT		:	Redo LAYERPREVIEW	1		Redo LAYERPREVIEW	1
:		: «Cancel»		:	Redo PASTE	1		Redo PASTE	1
: _DELETE		:		: _DELETE	Redo MOVE	1		Redo MOVE	1



Usability Test - Questionnaire

POI Editor - Usability Test

The goal of this test is to measure how well the POI Editor meets the minimal requirements and most common use cases. During this test you'll be given simple tasks to perform with the POI Editor. After completing the tasks there will be overall evaluation section where you can describe your overall experience and give feedback about the application

* Required

1. **Your expectations** *
   What are your expectations and thoughts towards the application?

Tasks

There are 7 tasks for you to complete. Describe your actions during the tasks and after completing each task please write down how you managed to complete it and how your experience was.

2. **1. Find and edit single POI** *
   There's POI in the map called Restaurant1. Customer would like change it to a cafe and rename the POI to a Café1. Your task is to find this POI and make the necessary changes to it.

3. **2. Add an area to a POI** *
   There's a room named Room1 which doesn't have area. Customer would like to have the area of the following the room's walls. Also the area colour should be the following: "#949494"

4. **3. Add a new POI** *
   Add POI indicating that Shop1 is under construction. The POI should cover all of the Shop1 and its area

5. **4. Edit multiple POIs** *
   Customer would like that all the toilet POIs would not have any text shown in the map. Also every toilet POI should have a tag "category_washroom".

6. **5. Delete multiple POIs** *
   In the basement there's three POIs indicating ATMs that need to be removed. Find a way to delete the POIs

7. **6. Find a missing POI** *
   There should be a POI (id: 59e723b06a6c990ec0408c3c) in the basement floor indicating where the stairs are but it's not shown in the map. Find the missing POI and make it visible.

8. **7. Make changes go live** *
   Find a way to make changes go live and go to a page

Evaluation

Describe and rate your experience with the application

9. **How was your overall experience?** *

10. **Usability** *
    On a scale 1 to 5 how usable was the application
    Mark only one oval.

	1	2	3	4	5	
Not usable at all	◯	◯	◯	◯	◯	Very usable

11. **What did you like the most?** *

12. **What did you like the least?** *

13. **Comparison** *

How do you find the application when considering current map maintenance process? Do you find it to be more efficient and usable?

Usability Test - Results

Timestamp	Your expectations	1. Find and edit single POI	2. Add an area to a POI	3. Add a new POI	4. Edit multiple POIs	5. Delete multiple POIs	6. Find a missing POI	7. Make changes go live	How was your overall experience?	Usability	What did you like the most?	What did you like the least?	Comparison
3/9/2018 14:59:16	Easy, fast, reliable way to manage points of interest and visual spaces within our indoor maps.	Finding the POI was straight forward in this case as the map was very small and it instantly stood out to me. Clicking the POI gave me the information needed to make the name change and switch the POI type to Cafe. The changes were made visible instantly by the text and icon changing. This gave an overall expected experience and enabled a quick and easy process for change.	The option to add an area was nice and clear once I clicked the Room1 POI. Adding the area was made simple with the use of snapping each corner to the walls as I dragged the points around. Changing the colour was as simple as copy and paste in the colour code in the selection tool.	It was not at first obvious to over the existing POI with the under construction POI and area. The style means it will be on top of the walls so it can be a challenge to make it align right.	Using select similar would be useful on larger buildings but as there were only a few in this case I selected manually. Updating the changes was as simple as it could be.	Select similar for just that floor made it very quick to delete all ATM pois	Search box found the POI instantly and switching the POI type was quick and simple.	The cloud upload and generate tiles button did the job. It should be made clear that you must re generate tiles for the changes to take effect.	Wonderful, smashing, great! This is a real game changer!	5	It didn't have any bugs	It would have been easier to get started with maybe a few more visual hints and tips. An intro or walk through etc.	Huge improvement from the current process.Time and effort in making changes and adding data is much faster and clear. Current methods would have taken more than double the amount of time to get to the same result. It is very likely that the current method would have not worked perfectly the first try as there are no instant visual feedback of the changes being published.
3/9/2018 16:06:37	I expect the POI Editor to speed up the process of making map changes and make it more user friendly	Found the restaurant from the map and clicked it. Then I changed its class from the dropdown menu and its name from the Attributes field. I also changed the identifier. Class dropdown menu should be extended so its easier to find the right choice.	I tried to click the room title but was not able to due to some bug. After trying to erroneously cover the room poi with a new one, I remembered to use the x-ray mode to manipulate the room poi and create an area for it. The colour was already se to the right one. X-ray mode is a really good thing to have.	I started to create a new poi until I remembered that I can just copy and paste the shops poi and area and just give it a new class. When pasting and object a clearer indicator should be given where the new poi is on the map.	I started selecting the toilet pois and deleting their names one by one. I could have used the search but because the map was unknown to me I opted to do it manually. I also added the new tag to all of the toilets.	I selected the ATMs using the alt+click command and deleted them all.	I used the search field and given id to find the missing poi. I then gave the poi a new class stairs to make it visible. When searching pois, option to center the view to the poi should be given.	I saved the changes and tried to generate the tiles. Two errors came up due to my created and deleted pois not actualy deleting and causing an error. I selected them and by using backspace command deleted them for good and resumed generating tiles.	Apart from few small bugs it is usable and does what was promised. I do think it will be an awesome tool after some updates.	3	Ease of use, clear user interface. I makes my work so much faster and easier.	Dropdown menu for class was too short. Short tutorial for all the features would be good for inexperienced users.	It makes the process much faster and more efficient.

Timestamp	Your expectations	1. Find and edit single POI	2. Add an area to a POI	3. Add a new POI	4. Edit multiple POIs	5. Delete multiple POIs	6. Find a missing POI	7. Make changes go live	How was your overall experience?	Usability	What did you like the most?	What did you like the least?	Comparison
3/13/2018 10:52:0	I expect that with the POI editor I can more easily modify map data than with the current process - that I can just click and edit items visually instead of writing API calls. Now, if I get a request to edit an attribute for some POI it's always an annoying and error-prone task. I also think that with the application customers can do a lot of things themselves that they previously needed our help for.	The POI was easy to find via the search. Title change was easy. I didn't at first notice the type of POI at the top of the screen and tried to edit the tags first, then wondered why the icon didn't change. I didn't know at first how to confirm the changes, though the confirm button is clear and had a helpful mouseover text.	I first tried to click the text but couldn't select it, so I selected the POI via search. Adding the area was clear. I first didn't know how to stop drawing, so I pressed Escape and it unselected the POI, leaving the area which was good. There could still be a text note on how to stop drawing. For some reason the area default color was 949494 (grey) and when the area was drawn I was not sure if it was somehow inactive, since it was grey - it would be good if the default color was bright.	I added a construction POI via the dropdown menu. First it was a bit unclear how to move an existing POI, but it did work as expected. Adding the area worked fine.	I searched for the POIs. I expected that just pressing Enter would start the search, but I had to click the icon. When viewing multiple POIs in the search results I expected that the result rows would have a checkbox if more than one can be selected at one time, though it is easier to select several by clicking the big area and not having to hit a small checkbox. I liked that the search results stay visible when clicking on one of them and it doesn't immediately zoom to the POI. Removing the title didn't work at first when I just tried to remove the text by pressing backspace in the title field. Still, it makes sense too to have to first type something. The color coding of the tags is not immediately clear though I can expect that a lighter colour means that not all POIs have that tag.	Selecting multiple POIs from the map wasn't clear at first, I tried cmd-click but it was alt click. After that deleting was easy though, the symbol was clear. The prompt text could be confusing for the user, instead of meta it could say database, backend, or just that the change is not final yet.	I found the POI by typing the id into the search, which was intuitive. At first the POI wasn't visible in the map view, but because it's a small building I knew where to search for it. If it was a larger building there might need to be a feature to zoom to the POI. Then I changed the POI type to stairs, and the icon appeared nicely.	The icon to upload changes is clear. The text could be more clear, the user might not know what meta service is or where the changes will go. I didn't know that I need to update the maps in a separate step, and even then 'generate tiles' might not be informative to the user. They might not know it means map tiles. After the tile generation has started, it could be good to have some kind of visual information that the map needs to be refreshed (even though it says so in the bottom bar). Maybe a minute after tile generation an icon could appear on the map reminding to refresh the page.	Overall the editor felt robust and clear. I didn't know what all icons meant but the icons had mouseover texts explaining their functionality. Changes were saved when you clicked elsewhere on the map so it didn't feel you could lose your work. Some things were a bit confusing still, like selecting multiple pois, the language (updating meta) and confirming the changes.	4	Search, selecting multiple POIs from the search. The responsive style changes (for example, changing the POI type changes its symbol). The application is easy to learn and feels powerful.	A lot of features were hard to discover, like the list of edited POIs, how to select multiple POIs, how to stop drawing. Sometimes clicking on a text didn't make it selectable.	It's definitely more efficient. Currently updating means many different steps (update cad - upload - generate tiles, or PUT edits to POI - generate visuals), with the editor it's just a few clicks.